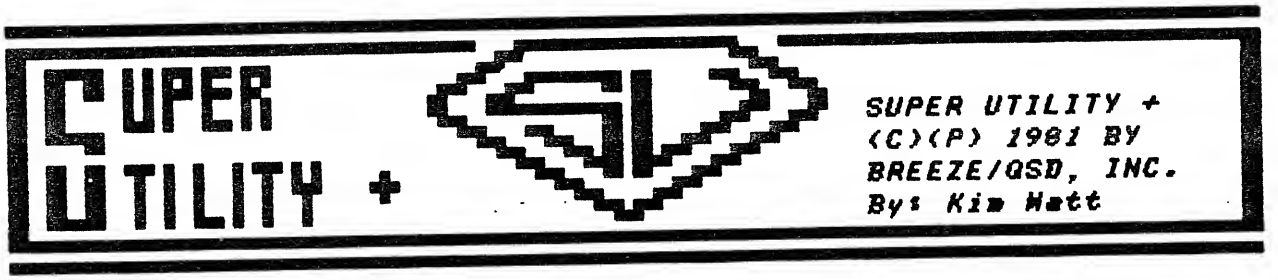
Inside SUPER UTILITY PLUS

BY

PAUL WIENER

WITH AN AFTERWORD BY KIM WATT



EDITED BY R B REYES, PH.D
AND DENNIS A. BRENT

POWERSOFT (tm)

11500 STEMMONS EXPRESSWAY. SUITE 125
DALLAS, TEXAS 75229
PHONE (214) 484-2976
MICRONET 70130,203

POWERSOFT IS A DIVISION OF BREEZE/QSD, INC.

=INSIDE SUPER UTILITY PLUS=

*** Table of Contents ***

Notes on Nomenclature

I've always been a disciple of Alice's caterpillar, making words mean what I mean them to mean by making them go where no similar words have gone before. Less intentionally, I have also been one of our century's leading exponents of creative spelling. My editors and friends, who have to interpret my day to day scribblings and utterings, will tell you to what extremes I've been known to carry these propensities.

Since this book is being published by a sane publisher, I've allowed Microproof to veto some of my more original concatenations of letters. With regard to inventing new words, using old words in strange ways, and unilaterally enacting new laws of grammar, I've abandoned departs of speech and restricted myself to the conventions--as much as possible. For instance, I'll be using the smaller-than and greater-than symbols ("<" and ">") as brackets to indicate keys which you are supposed to press. So when you see a statement like 'Press <1><ENTER>', you'll know you're intended to press the key that says "1" and the key that says "ENTER."

Sometimes I've used the more or less standard shorthand of saying "Enter <Y>" to mean press <Y> and then press <ENTER>. "Press," "type," or 'key <Y>', on the other hand, indicates that you're not supposed to press <ENTER> afterwards.

In some instances, I've allowed myself to stretch things just a little. I've used single quotes instead of double quotes to indicate something that Super Utility puts on the screen. So 'HIGHSPEED=Y' means that when you look at the screen, you will see HIGHSPEED=Y but no quotes.

But I've also used single quotes, rather than doubles, to escape a dilemma--the dilemma being that it's a grammatical requirement to enclose certain punctuation within quotes, even when logic and clarity require placing the punctuation outside the quotes. So I'll tell you to type '1,1,1', rather than "1,1,1." Notice how in the former, the third comma is outside the single quotes, but in the latter, the period is inside the double quotes where it creates the following ambiguity: do I or don't I want you to type the period? You'll notice that I've already used that escape hatch earlier in this preface.

In certain cases, I have gone ahead and coined new terms after all. I've worked up a taxonomy of disk errors. You'll learn about compliant errors and stubborn errors. You'll learn about two kinds of peccadilloes, and about nightmares. The less said about nightmares the better.

"Farkled" is a word you'll be seeing a lot of in this book. It's used by Jesse Bob Overholt in his Alternate Source columns. It's usually used in conjunction with munched disks. It means "messed up."

I sometimes use the term "TRSDOS III" when I get tired of spelling out "Model III TRSDOS."

A last word about words: this book is about Super Utility Plus. Much of the information simply doesn't apply to plain old Super Utility. But the manuscript looked so cluttered up with "Plus"'s that I pulled all of them out at the last minute. So when you see Super Utility, remember, I mean Super Utility Plus!

A last word: This book is not intended to be a substitute for your Super Utility instruction manual. My purpose is to fill in the background needed to use Super Utility effectively, describe some extended application techniques which aren't covered in the manual, and to clarify a few points which were covered, but have been generally misunderstood.

This book won't be very useful to you if you don't use your Super Utility manual as well. Please keep it handy and refer to it whenever necessary.

Chapter I
TECHNICAL INTRODUCTION

Though it's not strictly necessary, knowing a little about the way information is stored on and retrieved from your disks can help you understand Super Utility Plus and use it more effectively. All TRS-80 disk I/O takes place through a special piece of hardware manufactured by the Western Digital Corporation. It's called the Floppy Disk Controller chip (FDC). In this chapter I will describe some of the FDC's characteristics, functions, and peculiarities. Wherever appropriate, I will indicate which features of Super Utility Plus allow you to observe or control the FDC functions, or otherwise capitalize on their results. Detailed instructions on how to actually use Super Utility Plus to achieve these effects will be found in various other chapters.

Some of the information in this chapter may seem a little technical. If you're not into electronics or machine language, don't worry. You can probably follow this anyway. I don't know the difference between a casistor and a repacitor, but I can still relate to the concepts presented here. Even if you can't or don't want to acquaint yourself with these technicalities, it might be a good idea for you to skim this section anyway. The references to other chapters will help you zero in on the sections of this book which are more pertinent to your interests.

The FDC has the ability to translate back and forth between serial (bit by bit) data and parallel (byte by byte) data. This is necessary because the Z-80 handles data in parallel, but data must be moved to and from the disk serially. The FDC also contains what amounts to a dedicated microprocessor which performs certain control functions and computations that the TRS-80's Z-80 isn't fast enough for.

Though the FDC is a wonderful device, there are certain facets of its internal logic that can create difficulties, and there is no way around them other than convoluted programming. In this book, I will not try to explain why the FDC chip does things as it does. That will be accepted as part of the given. But where pertinent, I will explain what the FDC does, so you will understand why Super Utility Plus (and other software) has to be the way it is. This knowledge will also help you use Super Utility to the fullest advantage.

The division of disk space into tracks, granules, and sectors has been adequately dealt with in numerous popular books. Tandy's TRSDOS & Disk BASIC Reference Manual (that's your good ol' Model I TRSDOS users' manual) has a brief but

intelligible treatment of the topic in section 2, Mini Disk Operation. There are even a couple of nice diagrams. Page 74 of the Model III Disk Operator's Manual also mentions the subject, but there aren't any diagrams. If you would like more information about the subject matter of this chapter, see the informal bibliography near the chapter's end.

Briefly, the surface of a disk is divided up into concentric rings, called tracks. The oldest TRS-80 disk drives could only read and write to 35 such tracks. Most newer drives can handle at least 40, because the read/write head can move farther in toward the center of the disk. Both the 35 and 40 track drives use a 48 Tracks Per Inch (TPI) format.

Eighty-track drives are now becoming popular, but they access tracks which are only half as wide as those used by 35 and 40 track drives. Thus, disks used by 80 track drives may be said to have double track density, since they contain twice as many tracks per inch as do disks used by standard drives. This should not be confused with the more conventional type of double density, which puts more data in each track. (Editor's note: Some disk drive manufacturers have taken to using the term "quad density" which refers to a combination of both "double track density" and the conventional "double density"; hence a "quad density" drive is usually an 80-track drive capable of double-density)

Since a track written by an 80 track drive is only half as wide as one written by a forty track drive, 40 track drives can't read diskettes which were written by 80 trackers. However, 80 track drives can read disks created by 40 track drives. All the information is present and detectable. However, when the 80 track drive tries to step to another track, it won't move the head far enough to reach it. Therefore, when an 80 track drive is dealing with a 40 track disk, the system software must move the head twice, instead of once, to reach the next track. Super Utility's Double-step feature accomplishes this. You select it from the CONFIGURATION mode.

With Double-step, 80 track drives may read data written by 40 track drives, and they may do so fairly reliably. However, it is risky to attempt to write to 40 track diskettes in 80 track drives. If you use Super Utility's Double-step feature to write, you will probably succeed in creating data which can be read back--only while the target disk is still in the 80 track drive! If you intend to use the disk in a 40 track drive again, you may be heading for trouble. See the MISCELLANEOUS Chapter for a more detailed account of this topic.

Another caution: As stated earlier, standard 35 and 40 track drives use a 48 Track Per Inch (TPI) format. Eighty track drives normally use a 96 TPI format. This is why a Double-step feature works. One TPI is exactly double the other TPI, so moving the head twice gets to the same place. However some drives (especially some old Micropolis 77 trackers) use a 100

TPI format. Disks written on such drives will be incompatible with normal drives, with or without the Double-step feature.

The TRS-80 uses soft sectored diskettes, which means that there is only one index hole on each disk. This hole is detected by an optical sensor within the disk drive. It defines the starting point of each track for the system (see figure 1).

Tracks are divided into artifcial units called grans. In what I will refer to as The Standard Scheme (TSS), each track has two grans. Grans are further divided into units called sectors. In TSS, each gran has 5 sectors. Thus, each track contains 10 sectors (see figure 2). In TSS, each sector contains 256 bytes of user accessible information, plus some special data normally used only by the system. Super Utility allows you to monitor and alter all this information--both user and system.

Other arrangements are possible. Non-standard, or "protected" disks can vary these modes of disk organization, as well as several other factors which will be described later. Even some "standard" disks may not conform to the picture just painted. For instance, some double density disks (DOSPLUS, LDOS) have six sectors per gran and three grans per track while others (TRSDOS III) have six grans of three sectors each per track. NEWDOS-80 and DBLDOS use lumps instead of grans. Lumps are explained in the MISCELLANEOUS Chapter.

As I insinuated a couple of paragraphs back, a disk contains a certain amount of information other than "user data." The bulk of this data is placed on the disk during the formatting process. Mostly, it consists of special gaps and ID fields which separate and identify the various tracks and sectors for the FDC.

Consider, for a moment, what a remarkable task the Floppy Disk Controller performs as it reads information from a disk. As you probably know, the data is recorded on disk as a series of tiny magnetized pulses. As the disk spins, over 100,000 of these dots fly by the disk drive's read/write head every second. To interpret the data properly, the FDC must maintain perfect synchronization with the flow of pulses . Without perfect sync, bits could be lost. The loss of a single bit would cause all further data to be out of place, or "shifted", in memory. Such shifted data would be useless, as the computer would interpret it incorrectly.

To help achieve correct synchronization, the FDC has its own internal clock. In the TRS-80, the FDC clock runs at 1 mhz. With the help of its clock, the FDC puts a series of evenly spaced pulses, called clock bits, on the disk. When writing data to the disk, a one bit is represented by placing a data pulse between two successive clock bits. A zero is represented by leaving the interval between two clock bits blank.

When the system reads back the information, the FDC simply locks into the flow of clock bits. This lets the FDC look at the slot between each successive pair of clock bits and interpret

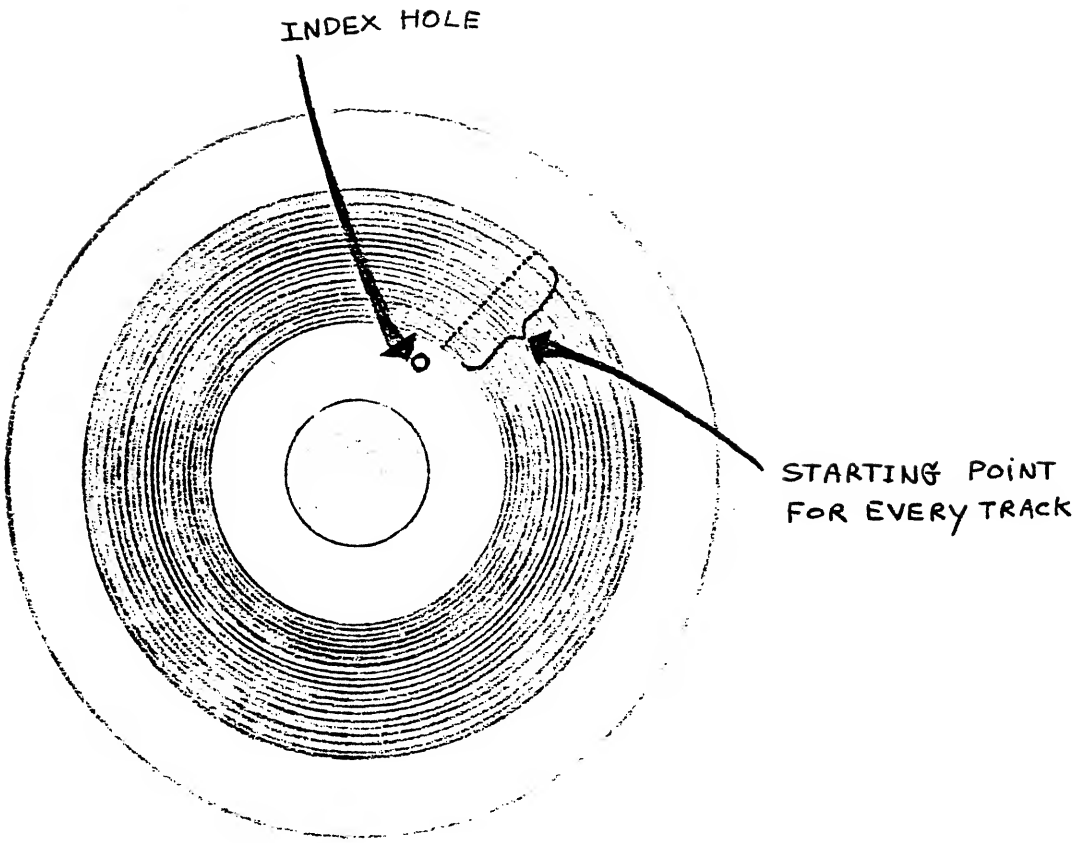


FIGURE 1

blank slots as zeros, and slots with pulses as ones. Actually, it's not quite that simple, but almost. An FM (Frequency Modulation) method is used, since the frequency of the bit flow will be higher where there is a data bit between clock bits than where there isn't.

You might think that all this would assure perfect synchronization. Unfortunately, it doesn't, largely because of small variations in the speed of spinning disks. Such variations can nullify the high precision of the FDC's clock, so that in effect, the FDC which wrote the data was timed differently from the FDC which tries to read in the data, even if it's one and the same FDC!

Another cause of trouble is that the inner tracks are physically shorter than, though logically equal in length to, the outer tracks. In other words, on the inner tracks, the same number of clock and data bits must be jammed into a shorter band. This can make it more difficult for the FDC to stay in sync and separate the clock bits from the data bits.

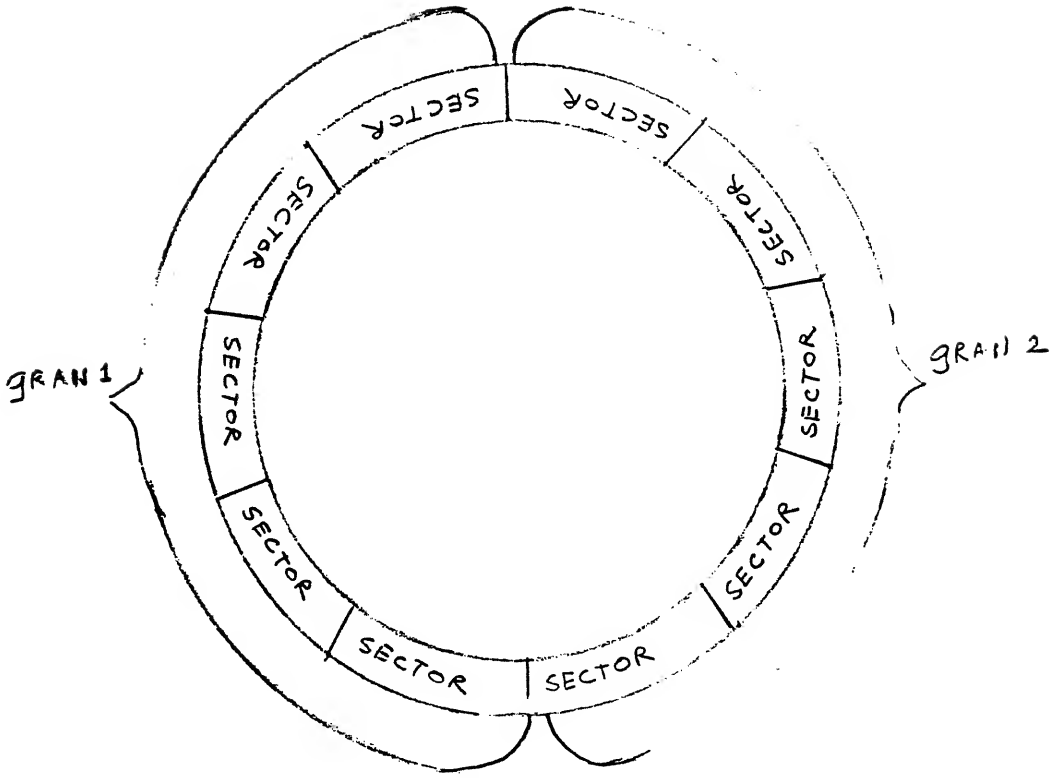
As you've probably heard, data separators are available to cure these problems. An important part of their read circuitry is a phase locked loop widget which dynamically adjusts the clock rate of the FDC to match the flow of clock bits on the target disk. Such external data separators are quite effective. Yet, even with one (and especially without one), it is still possible for errors to occur. Consequently, certain other measures are taken to insure accuracy.

Every sector on the disk has a special ID header. This header is preceded by a "gap" which separates it from the previous sector. The sequence of gap and ID header makes it easy for the FDC to lock into the bit flow for that sector. Obviously, the longer the sector, the more likely the FDC is to drift out of sync during a read. Thus, it is important to keep the sectors sufficiently short to ensure positive sync.

The people who devised the original TRS-80 system software apparently felt that dividing a track into 10 sectors, resulted in short enough sectors (256 bytes a piece) to minimize errors. Almost all other TRS-80 system software programmers have followed suit.

Super Utility allows you to create special disk formats. If you wish, you can create disks with fewer (or more than) than 10 sectors per track. You may even fill an entire track with only one giant sector. If you do so, however, you should be aware that you're risking a higher probability of error. You can create these unusual tracks with super Utility's SPECIAL FORMAT, or BUILD FORMAT TRACK options.

As you have seen, a major purpose of dividing each track up into sectors is to give the FDC frequent opportunities to resynchronize. In order for this scheme to work, each sector must start in a way which the FDC can easily detect and lock onto. Here's how that is arranged.



2 GRANS PER TRACK
5 SECTORS PER GRAN
10 SECTORS PER TRACK

FIGURE 2

Every sector contains two subfields which I shall refer to as the header subfield and the data subfield (see figure 3). The header subfield (sometimes called the sector ID block or ID pack) is a seven byte segment that starts the sector (Editor's note: this is not the same as the "Pack ID" referred to by TRSDOS 2.3's BACKUP utility when you attempt to do a backup to a previously formatted disk with a different name. That "Pack ID" refers to the diskette's name/master password combo). It consists of a one byte address mark followed by four one byte designators and a two byte CRC. The address mark is always FE hex. The header address mark is also called an ID address mark. Each of the four designators contains pertinent information about that sector--namely the track number, head number, sector number, and sector length.

Notice that there is no separate identifier for the track as a whole--the track number is specified by the first designator in every one of its sectors. The head designator is useful with double-sided diskettes, or hard disks with more than one track per "cylinder." On all "normal" TRS-80 diskettes, the head designator should contain zero (0). And while we're on the subject, it should be noted that Super Utility Plus does not offer any support for double-sided drives unless each side has been wired as to be visible to the system as separate drives.

The sector-number designator is self explanatory. The sector length designator contains a code that indicates the number of data bytes in the sector. With standard TRS-80 disks, it will always be a 1, indicating 256 bytes of user data. This is what is sometimes referred to as "IBM format." In the IBM scheme of things, a length byte of 0 indicates that there are 128 bytes of data in the sector, while a length of 1 indicates 256 data bytes. In the "non-IBM" convention, the length byte multiplied by 16 gives the actual number of data bytes in the following sector; however, a length byte of 0 indicates that there are 4096 bytes to follow. The header field is put on the disk when the disk is formatted, and is never altered by the system--unless the disk is reformatted.

The second, or data, subfield comprises the main body of the sector. It contains the actual user data. The data subfield is rewritten every time data is saved to that sector.

Each of the two subfields is preceded by a "gap", which typically consists of 12 FF hex bytes followed by six zero bytes. I use the word "typically" advisedly, because there is a great deal of variation from this norm, both with respect to the number of bytes in the gap and the values of those bytes.

The data subfield, like the header subfield, starts with a one byte "address mark." When these address marks are written to disk, the frequency of the accompanying clock signal is changed by dropping some of the pulses. This makes the address marks unique. Therefore, the FDC can easily and quickly find address marks, and will not be fooled by sequences of data marks which mimic address marks (the data can't fake the change in the clock

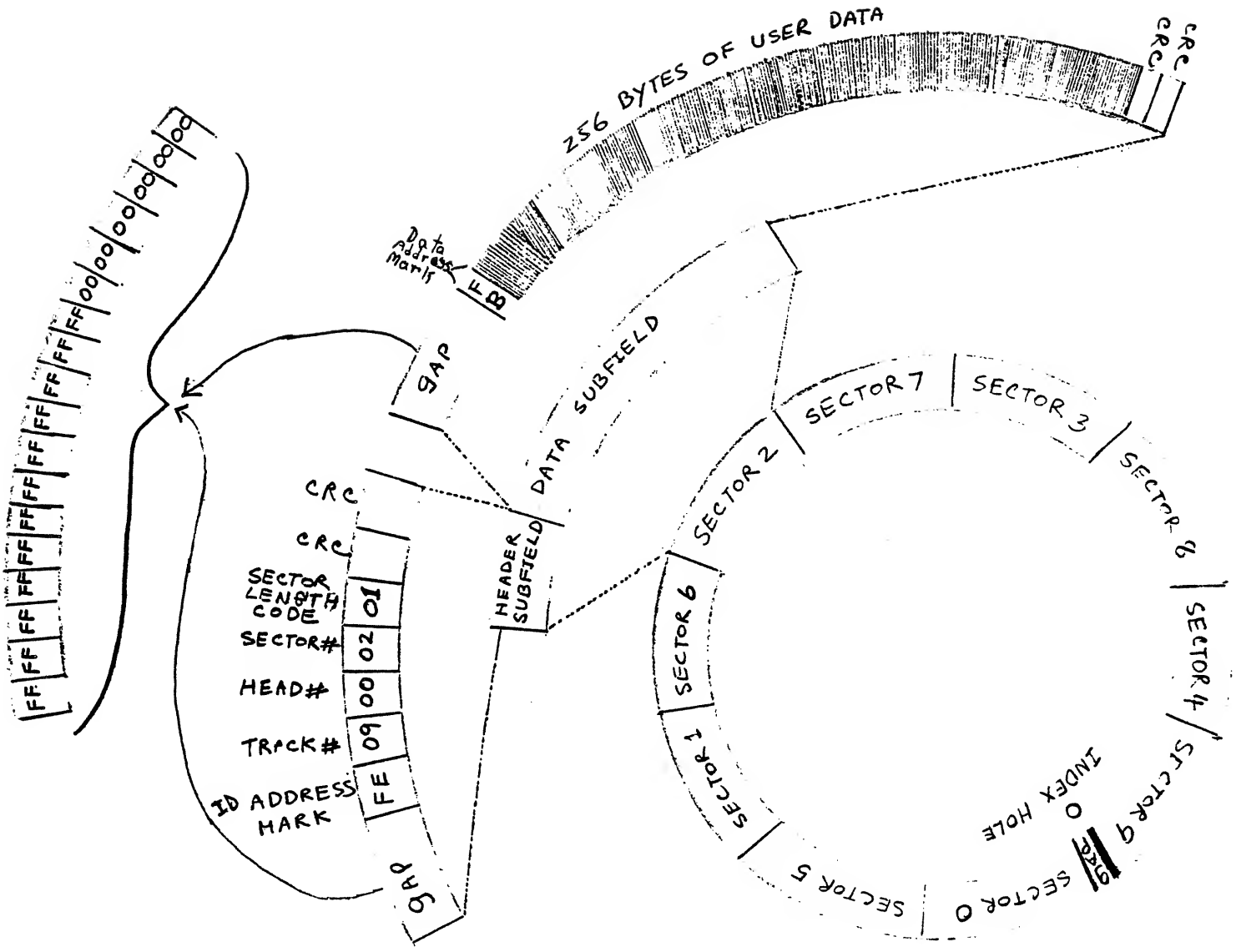


FIGURE 3

pulses).

For header subfields, the address mark used is FE hex. The address mark used for the data subfield is often referred to as the Data Address Mark (or DAM). The actual byte used varies, depending upon whether a Model I or Model III is being used, whether double or single density is employed, what operating system is in use, and whether the sector in question is a normal one or part of the disk directory. To simplify this discussion, we'll assume a garden variety situation. This means a Model I running TRSDOS in single density. For other situations, see figure (4). The DAM for this "garden variety situation" is FB hex for standard sectors, or FA hex for directory sectors. The directory is given a different DAM to help the disk operating system in locating it.

Both the header and data subfields end with a two byte Cyclic Redundancy Check (CRC). CRC's are used to detect errors which occur despite all precautions. A CRC performs much the same function as a parity check or checksum, but is much more reliable. Essentially, it is a 16 bit number derived by applying a polynomial evaluation to every bit in the associated subfield. In other words, the CRC at the end of any subfield is derived by an operation on all the other bits in that subfield. This computation is one of the super-fast ones that the Z-80 leaves to the dedicated FDC microprocessor.

When the FDC reads back a sector, it recomputes the CRC bytes and compares them to those that were written on the disk during the last formatting or data-save operation. Any disagreement causes the FDC to flag an error (some disk operating systems respond to this by displaying "parity error" messages although, strictly speaking, what has occurred is not a parity error but a CRC error -- Ed.).

A standard disk track consists of ten sectors, separated by gaps, as described. In addition, there is an extra gap between the track's starting point, as defined by the disk's index hole, and the beginning the first sector. This gap is similar to the others, but normally a little longer.

Super Utility's ZAP package contains options to READ ID ADDRESS MARKS and ALTER DATA ADDRESS MARKS. These give you good control over the address marks on your disks. READ ID ADDRESS MARKS also can display CRC values as read from disk, and tell you whether the currently calculated CRC's agree with those read. Also, super Utility's CONFIGURE SYSTEM option ensures that Super Utility's own use of address marks will be consistent with that of the operating system of any target disk.

As I said, each sector header subfield contains a track and sector number. It is possible to give any sector a "false" ID, simply by putting nonstandard information in its header. Thus, a sector may be the first of ten physical sectors on the first physical track on a 40 track disk, and yet have a header that says it is the 108th sector on the 99th track.

Figure 4

Directory DAM Chart

DOS type	Model I Sgl. Den	Model I Dbl. Den	Model III Sgl. Den	Model III Dbl. Den
TRSDOS I	A9		A1	
TRSDOS III		A8		A0
LDOS I(S)	AB		A1	
LDOS I(D)	A9		A1	
LDOS III(S)	AB		A1	
LDOS III(D)		A9		A1
ALL OTHERS				
I(S)	A9		A1	
I(D)		A9		A1
III(S)		A9		A1
III(D)		A9		A1

In fact, DOS never labels physically adjacent sectors consecutively. For better disk I/O, it uses the following pattern: 0,5,1,6,2,7,3,8,4,9. As you can see, this is composed of an interleaving of the two following patterns: 0,1,2,3,4, and 5,6,7,8,9. Since such an arrangement is normal, it is rarely referred to as false labeling.

Tracks whose numbers don't reflect their actual physical position on the disk, however, are another matter. Tracks may not only be mislabeled, but each one of a track's sectors may be given a different track number! This is why in some displays, Super Utility shows two track numbers. In the ZAP module, they're labeled TRACK and TRUE. TRUE shows the track number as read from the disk; TRACK shows the number of the track according to its physical location. In Read ID Address Marks, SOURCE shows the actual physical track number, while TRACK shows the track number as read from the disk.

As you can see, the designers of the FDC gave a lot of thought to error prevention and detection. In the DISK ERRORS Chapter, we will see what kinds of disk errors can occur, and how to detect and correct them.

INFORMAL BIBLIOGRAPHY

TRS-80 Disk and Other Mysteries, by Harvard Pennington, published by IJG, seems to be considered the classic in the field. I understand an updated version is being prepared which will have information on double density, Newdos 80, and other new mysteries. So if you're thinking of buying TRS-80 Disk and Other Mysteries, it may be worth your while to wait for the update.

If you can get by with a straightforward presentation of the subject, you might look into book 2 in the Mystery series, Microsoft BASIC and Decoded and Other Mysteries, by James Farvour. It's an information, rather than explanation, oriented book. Its ten or so pages on disk I/O present a concise recap of much of the information in Harv's book, with some new data to boot.

Other good books to have handy are Barden's Disk Interfacing Guide and Pathways Through the ROM. Pathways is anthologized and edited by George Blank, and contains material by Robert Richardson, Roger Fuller, John Phillips, George Blank, John Hartford (not, as far as I know, the singer), and the Western Digital Corporation. It includes Supermap (a comprehensive TRS-80 Model 1 memory map), the technical manual for the 1771 (the Model 1's FDC), and an essay on DOS containing practical information which I haven't seen published anywhere else.

An excellent description of Model I TRSDOS formatting is presented in John Brule's Some Disk Considerations, in The Alternate Source, Volume II, number 3. Also, the LDOS Quarterly, volume 1, number 1, contains an enlightening article on DAM's by Roy Soltoff.

For more technical information on Super Utility Plus, refer to the SU+ TECH MANUAL, published by Breeze/QSD, Inc. It is written by Kim Watt, with DOS notes by Pete Carr.

Chapter II
CONFIGURATION

You get into the configuration module by entering <9> from the main menu. As soon as you do so, the screen will be filled with a display called the configuration table. Configuration may seem complicated to some new users, but it is actually very simple. It's a way of telling Super Utility about factors which can vary from use to use.

Some factors may be different from disk to disk, and others from computer system to computer system. Those which vary from disk to disk include the DOS contained on the disk, the number of formatted tracks, and the density (single or double) of the disk's data.

Factors which may differ from system to system include the characteristics of your printer (if you have one), the absence or presence of a high speed modification, the number of disk drives you have, and the speed characteristics of each.

Super Utility needs to know whether your printer is serial or parallel, if it needs a line feed after each carriage return, and if it's capable of printing lower case letters and/or TRS-80 graphics. If you printer handles graphics, Super Utility needs to know whether it's an Epson MX-80 or not (MX-80's may require the computer to adjust the graphics characters before sending them to the printer).

If you have a high speed mod, Super Utility wants to know three things. 1) whether you want high speed turned on or off; 2) how to turn it on; and 3) how to turn it off.

If you have an 80 track disk-drive, and want to use it with disks that were formatted on a 35 or 40 track drive, Super Utility needs to know about that, too. Naturally, this information is entered via the configuration module.

You can also use the configuration module to inform Super Utility that you want a drive software write-protected. This affords similar protection to that obtained by putting a write protect tab on a disk. Once you've told Super Utility to write protect a drive, it will refuse all requests to write to that drive until you cancel the write protect status.

Figure 10 depicts a typical CONFIGURATION table. The configuration shown is similar, but not identical, to the one in the Super Utility manual. It's possible to change most of the data in the table, simply by typing in different data. We'll get to that shortly. But first, lets analyze the information on line one of this sample:

```
=>HISPEED=Y, DUAL=N, SAVE=N, ON=3E01D3FE0000, OFF=3E00D3FE0000
```

The "=>" at the beginning of the line means that this is the "current" line--in other words, this is the line which may now be altered by typing in new data. Note that there are 5 items of information, separated by commas, on this line:

1 HIGHSPEED=Y
2 DUAL=N
3 SAVE=N
4 ON=3E01D3FE0000
5 OFF=3E00D3FE0000

Items 1,4, and 5 have to do with controlling any high speed modification which may have been installed in your computer. Let's consider the meaning of each.

I guess I'd better preface my discussion of speed configuration with this comment: If you have a normal TRS-80, with no high speed clock or similar modification, it doesn't matter much how you configure items 1, 4, and 5. But you should keep item one set to 'HIGHSPEED=N'.

'HIGHSPEED=Y' indicates that your computer has a high speed mod and you want Super Utility to keep high speed turned on. Here's why Super Utility may needs to know you're running high speed. If your computer is in high gear during disk I/O, the timing may be thrown off enough to cause errors. Some "intelligent" high speed mods automatically restore normal speed during disk I/O and then go back into high as soon as the I/O is finished. But some do not. If yours doesn't, Super Utility can adjust its timing loops to secure reliable I/O. It will also adjust timing loops controlling cursor flash and key repeat.

High speed mods usually have software commands which you can use to change speeds. Super Utility wants to know what those commands are for your system. Once it knows them, you can turn the high speed on and off from the key board by configuring 'HISPEED=Y' or 'HISPEED=N'. If you're not sure what the on and off commands are for your system, refer to your speed mod documentation .

Most of the popular high speed mods are controlled by output to port 254 decimal. In the current example, the high speed in our hypothetical computer is turned on by outputting one (1) to port 254. It's turned off by outputting zero (0) to port 254. If you were controlling the speed from BASIC, you'd enter OUT 254,1 to turn on high speed, and OUT 254,0 to turn it off. Super Utility, of course, uses Z-80 machine language rather than of BASIC. In Z-80 code, there are many ways to send a value to a port. The sequence 3E01D3FE is one of them. Here is a disassembly of the sequence.

Figure 10

```
=>HISPEED=Y, DUAL=N, SAVE=N, ON=3E01D3FE0000, OFF=3E00D3FE0000
  PRINTER GRAPHICS=Y, LOCASE=Y, MX80=N, PARALLEL=Y, LFEEDS=N

  +:0A= 40, TKS= 40, DIR= 17, STEP=3, DELAY=2/2, WP=N
            DENSITY=S, L-GRANS, TRACK0=S, SS=0, DAM=0
  +:1B= 80  TKS= 80, DIR= 40, STEP=0, DELAY=1/2, WP=N
            DENSITY=D, L-GRANS, TRACK0=D, SS=1, DAM=0
  =:2C= 35  TKS= 35, DIR= 17, STEP=0, DELAY=1/2, WP=Y
            DENSITY=S  L-GRANS, TRACK0=S, SS=0, DAM=1
  -:3I= 35  TKS= 63, DIR= 17, STEP=3, DELAY=2/2, WP=N
            DENSITY=D, I-GRANS, TRACK0=S, SS=0, DAM=0
.............................................................
```

```
3E01          LD        A,1
D3FE          OUT       0FEH,A
```

In the first line 3E says to load the A register. 01 is the the value with which it's to be loaded.

In the second line, D3 indicates that the value in the A register is to be sent out to a port. The FE means that the port to be used is FE (hex) or 254 (decimal). So executing the machine instructions 3E01D3FE results in a one being sent to port 254.

Now, let's take another look at 'ON=3E01D3FE0000'. It consists of six hexadecimal bytes. The first four, 3E 01 D3 FE is the sequence of machine code which we just looked at. It sends a one to port 254. The last two bytes, 00 00, are NOPs (pronounced either no-ops, or so as to rhyme with pops), or Z-80 code do-nothings. They are there because Super Utility accepts a 6 byte sequence to control a high speed mod, but in this case only four are necessary. Next, let's look at 'OFF=3E00D3FE0000'. Notice that the OFF sequence is exactly the same as the ON sequence, except that the 01 byte after the 3E has been replaced with a 00 byte. This specifies that high speed is to be turned off by outputting a zero to port 254.

If your high speed mod requires different instruction sequences to turn it on and off, you must determine what they are. If you have no knowledge of machine language, this may seem intimidating, but in most cases it's easy. Port output is almost always used. The only varying factor is the number of the port and the values to be sent to that port.

If your high speed mod requires a value other than one (01) to turn it on, simply substitute that value for the 01 in 'ON=3E01.....'. If a byte other than zero (0) turns off your high speed, just replace the 00 in 'OFF=3E00.....' with the byte that slows down your system. Always remember to use a two digit hexadecimal byte.

If you need to use a port other than FEH (254 decimal), substitute the port number for the 'FE's in 'ON=3E01D3FE....' and 'OFF=3E00D3FE....'. Again, remember to use two digit hexadecimal bytes to represent the port numbers.

If you have a high speed mod whose on and off instructions are not similar to the ones in this example, consult the high speed mod documentation to learn just what machine code sequence is needed to do the job. Remember, Super Utility Plus allows you up to six bytes for each instruction sequence. As you have seen, that's more than enough for what's typically required.

By the way, some people have found that on their systems, Super Utility works fine at high speed, even without adjusting timing loops to compensate for the speed-up. So it may be worth your while to experiment and find out what works for you. We'll get back to this in a few paragraphs.

Now that we've got most of the high-speed/low-speed stuff out of the way, let's go back and cover what we skipped:

'DUAL=N'. Super Utility allows you to do a screen print at any time by pressing <SHIFT><CLEAR>. But some times, when using Super Utility you may want a printed record of all the information Super Utility presents to the screen. In such cases, turning on the DUAL mode is more convenient than frequent screen printing. The effect of DUAL is similar to the Model III's DUAL command, or LDOS's LINK *PR TO *DO statement. It causes all significant screen output to go to the printer as well. In Super Utility's configuration table, 'DUAL=Y' means DUAL is turned on (simultaneous screen and printer output) and 'DUAL=N' means that dual is turned off.

'SAVE=N': Soon you will see how to use CONFIGURATION to tell Super Utility about your disks and disk drives. Sometimes you will have Super Utility configured to read a certain kind of disk in a specific drive--for example a 35 track Model I TRSDOS disk in drive 1. During the course of your work, you may want to put a different disk into that drive--say a 40 track model III LDOS disk. Super Utility has certain override commands (See page 5 of your Super Utility manual for a list) which make it convenient to perform such switches without going back to the configuration mode. The use of these overrides can alter the configuration table automatically. So can certain other Super Utility functions.

If SAVE=Y, you may use overrides as often as you like, yet Super Utility will always revert to the configuration you specifically selected in the configuration module. This is quite handy if you nearly always use the same kind of disk in each drive, with only a quick swap now and then. But it's inconvenient if you intend to work with the "swapped-in" disk for a while. Then you would have to use an override command with almost every Super Utility operation, or go back to the Configuration Module to reconfigure.

For instance, suppose SAVE=Y and drive 0 is configured for a single density TRSDOS disk. But you decide to use drive 0 to do some zapping on a double density LDOS disk. You want to read track 5, sector 0. You tell Super Utility to do so, using the proper override command. A minute later, you want to read track 10, sector 1 of the same disk. You will have to repeat the override command. Otherwise, Super Utility, which has SAVEd the single density TRSDOS configuration, will be unable to read the double density LDOS disk. To avoid entering repetitious overrides, set SAVE=N. This is the default mode when Super Utility is delivered.

If SAVE=N, every time you use an override, the new information about the current disk is passed back to the configuration module and automatically entered there for you. For example, suppose you use CONFIGURATION to set up drive 0 for a single density TRSDOS disk. After you have done so, the drive

zero line of the configuration display should start with something like this:

+0A=35,

The A indicates single density Model I TRSDOS.

Later you use an override command to read a double density LDOS disk. After reading the LDOS disk, if you go back to the configuration mode, you will see that the line for drive zero now starts as follows:

0D=35,

The A has changed to a D to reflect that a double density LDOS disk was the last disk read in drive zero. You may now perform further operations on the LDOS disk without re-entering the override command.

Sometimes, when you go to the configuration display, it may have changed, even though you haven't used any override commands. This can happen because Super Utility's automatic density recognition is also capable of updating the configuration information when SAVE=N. Also, certain operations which involve a disk's directory are capable of passing a new track-count back to the configuration module. Because of the higher degree of automation present in the 'SAVE=N' mode, we recommend that you normally keep SAVE set to N.

Sometimes, as a result of an automatic configuration change, the DOS specifier in the configuration table will be replaced with an asterisk ("*"). "*" means "Unknown DOS." In other words, an asterisk in the configuration table indicates that Super Utility is no longer certain of the disk's resident operating system.

To change an entry in the configuration table, the "=>" pointer must be pointing at the line you want to change. When you first enter the configuration module, the arrow will be pointing at the first line--the one we just discussed. To alter the line, simply retype it.

At the bottom, of the configuration table is a question mark ("?") followed by a dotted line. As you type the new configuration entry into Super Utility, your keystrokes will be echoed on the dotted line, so you can be be sure to spot any typing error. Correct errors by backspacing and retyping.

Notice that you only type the variables. In other words, you don't type "HIGHSPEED=N", only "N,". Here's a line that would change each entry in the first line of our sample:

N,Y,Y,3E02D3010000,3E03D3010000

This will result in the following update of the configuration table:

HIGHSPEED=N, DUAL=Y, SAVE=Y, ON=3E02D3010000, OFF=3E03D3010000

This says that the high speed mod (if any) should be kept off, the DUAL mode is turned on, and that the current configuration should not be replaced by any overrides that occur during use. The high speed on and off command sequences have been altered as well.

Let's see what the new high speed on/off sequences say. The "on" command is the same as the previous one, except that it uses port # one instead of 254, and sends a two instead of a one. The "off" command likewise uses port one. It sends a three to that port.

Now suppose you want Super Utility to start using high speed. Instead of re-typing the whole command line, you can go to the configuration module and enter a single "Y". Super Utility will display 'HISPEED=Y' at the beginning of the configuration table, and leave the rest of the line unaltered. In other words, when changing the configuration table, you need only type from the beginning of the line up through the part you're going to change. The rest of the line may be omitted.

Always enter six full hex digits when typing the highspeed "on" and "off" commands. If necessary, pad the right with NOP's ("00"'s), as in the examples.

To change any other line of the configuration table, press <ENTER> until the "=>" character is pointing at the line you want to alter. Then retype the line and press <ENTER>. If you entered the data correctly, the line will be updated and the "=>" will point at the next line. If your attempted input was illegal, the "=>" will remain where it was and the line will not be updated past the point of the error.

To go back to a higher line on the page, press <BREAK>, which takes you back to the top of the table.

If you have a high speed modification, now you may want to try the experiment I suggested a few paragraphs back--namely getting Super Utility to run at high speed, without adjusting its timing loops. To do so, lie a little! Re-enter line one of the configuration table. But for the 'OFF=....' sequence, type in the set of instructions that actually turns your high speed on. Also, tell Super Utility to turn off high speed by setting the configuration table to 'HISPEED=N'.

Super Utility will think it has turned off your high speed clock, so it will not use its special timing loops. But in reality, it will have turned high speed on. Experiment (on a non-crucial disk, of course!) with various kinds of disk I/O. Also, test the timing of the key stroke auto-repeat in the Zap and Memory Modify modes. If all goes well, you may want to make this the normal mode of operation.

Now for the configuration table's second line:

PRINTER GRAPHICS=Y, LOCASE=Y, MX80=N, PARALLEL=Y, LFEEDS=N

The word 'PRINTER' at the beginning of this line is a header indicating that all the items here deal with your printer. 'GRAPHICS=Y' indicates that it's capable of printing TRS-80 graphics. If your printer doesn't have that capability, change the "Y" to an "N" and Super Utility will send dots to your printer, instead of graphics characters.

'LOCASE=Y' means that your printer can print lower case letters. If you change this to "N," Super Utility will convert all lower case to upper before sending it to the printer.

'MX80=N' means that your printer is not an Epson MX-80. However, even if it is an MX-80, you may want to keep this configuration set to 'N'. Here's the story. Use "Y" only if your MX80 has its own TRS-80 configuration switch set to the TRS-80 position. Super Utility will then "adjust" graphics characters before sending them to your printer. If your MX-80 doesn't have its TRS-80 switch set to the TRS-80 position, the Epson responds like a "normal" printer, and Super Utility doesn't need to make any special adjustment.

'PARALLEL=Y' indicates that you have a parallel printer. If you change this to no, Super Utility will use a TRS-232 serial printer driver. NOTE: This is not a standard RS-232 driver. It supports the Small Systems Software TRS-232 serial interface and compatible devices. All printer data is sent out the cassette port, not the regular RS-232 ports.

'LFEEDS=N' indicates that your printer doesn't need a linefeed character sent after each carriage return. Most printers which can be used with a TRS-80 don't. If yours does, but Super Utility is configured to 'LFEEDS=N', then your printer will tend not to linefeed when it should, printing over the same part of the paper repeatedly. On the other hand, if your printer does not require the extra linefeed, and you have Super Utility configured to 'LFEEDS=Y', your printer will tend to skip lines and double space when it shouldn't. So if you have any doubt as to the linefeed requirements of your printer, just do a few experimental screen prints.

The rest of the configuration table refers to any disk drives which may be on your system. You can have up to a maximum of four drives. If you have fewer, there will still be four entries in the table. Soon you will see how to adjust the configuration table to let SU know how many (and which) drives are actually in the system.

Let's look at the sample entry for drive zero:

```
+:0A= 40, TKS= 40, DIR= 17, STEP=3, DELAY=2/2, WP=N
          DENSITY=S, L-GRANS, TRACK0=S, SS=0, DAM=0
```

It is composed of two lines. The first line, with one exception, contains information which you may alter directly by retyping the line. The second line contains "implied" information. It is derived from the data in the first line. The second line may be changed only by re-entering the first.

The first character on line one may be either a plus, minus, or an equal sign ("+", "-", or "="). "+" indicates that the device exists and is "logged into the system" (in other words, Super Utility knows it is there). "-" indicates that, at least as far as Super Utility is concerned, the drive is not in the system. "=" indicates that the drive is in the system, and is in the "double-step" mode. The double-step mode is for an 80 track drive which will be used to read a diskette formatted on a 35 or 40 track drive. Drives so configured will step twice, instead of once, for every track-step.

There are restrictions to be observed when using the double-step mode. In general, it's O.K. to read in this mode, but inadvisable to write. For fuller details, see the MISCELLANEOUS Chapter.

The second and third characters on the first disk line are a colon followed by a digit from zero to three. This simply identifies the drive number.

The fourth character is a letter from A to J. This declares three pieces of information about the diskette in the drive: 1) the TRS-80 model that it's for (I or III), 2) the density of the data on the disk (single or double), and 3) the operating system that formatted the disk (TRSDOS, LDOS, DOSPLUS, DBLDOS, or NEWDOS 80).

Page 5 of your Super Utility Plus owner's manual has a table showing which letter declares what information. There are a few DOS's not mentioned in the table. Use "A" for Newdos 2.1 (or Newdos Plus), Ultrados, and Model I single density Multidos. Also use "A" for single density Newdos-80. Use "F" for Model I double density Multidos disks, but don't try to use the Super Utility "Repair GAT Sector" feature on Multidos. Finally, use "B" for Model III Multidos disks. Also, if you have any NEWDOS 2.1 disks which have been modified with Doublezap to make them double density, use "I" for them.

Here is an updated list:

A = Single Density Mod I TRSDOS, Newdos 2.1 (also known as Newdos+), Ultrados, and Mod I Single Density MULTIDOS.
B = Mod III TRSDOS or MULTIDOS
C = Mod I or III Single Density LDOS
D = Mod I or III Double Density LDOS
E = Mod I or III Single Density DOSPLUS
F = Mod I Double Density DOSPLUS or MULTIDOS
G = Mod III Double Density DOSPLUS
H = Mod I Double Density DBLDOS (Percom)
I = Mod I or III Newdos/80 ver.2 Double Density with Single Density TRACK 0. (Also use "I" for Newdos 2.1 DD zapped by "DoubleZAP" for DD operation.)
J = Mod I or III Newdos/80 ver.2 Double Density with Double Density TRACK 0.

O.K. Let's focus on the first four characters for the first drive in our sample configuration table:

+:0A

The "+:0" says that there is a drive zero (0) in the system and that it's not set to "double-step." The "A" says that the operating system on the disk is Model I TRSDOS, or a member of the Model I TRSDOS family (NEWDOS 2.1, ULTRADOS, single density NEWDOS-80, or single density model I MULTIDOS).

The next two characters on drive zero's first line are an equal sign and a space ("= "). These are visual formatters and never change for any of the drives. Then comes a two digit decimal number which tells you how many physical tracks are present on the diskette in drive zero. In our example, the diskette is formatted to 40 tracks.

Then comes "TKS= " followed by a two digit number. This is the only datum on line one which is "implied," and therefore can't be altered directly. It shows the relative track count for the diskette. The number of relative tracks will normally be equal to the number of physical tracks. The exception occurs with NEWDOS-80 and DBLDOS, which use Disk Relative Sectors (DRS). I will present more information on DRS in the MISCELLANEOUS Chapter. Since the disk in our example is Model I TRSDOS, the number or relative tracks equals the number of physical tracks. Hence "TKS= 40."

"DIR= nn" shows the number of the disk's directory track. On our sample disk, the directory is on track 17. When working with NEWDOS-80 or DBLDOS disks, it will show the relative, rather than the real, track number.

"STEP= " and "DELAY= " pertain to hardware timing constraints of the disk drive. The stepping time refers to the time it takes the read/write head to move from one track to the next. Some drives must be allowed 40 ms (milliseconds) per step. Others can get by with only 6 ms. Most drives can step within at most 20 ms. Super Utility gives you a choice of four stepping

speeds per drive: 0=6 ms, 1=12 ms, 2=20 ms, and 3=40 ms. Our sample drive zero is set for "2," or 20 ms.

Please make note of two facts:

1) If a drive is configured to a single density DOS, it will not step faster than once every 12 ms, even if it's configured to the 6 ms speed. This is due to TRS-80 hardware design, and is not a bug in Super Utility.

2) It will never do any harm to configure a drive for too slow a step-rate, except that disk operations will lag a bit. However, configuring too high a speed can result in extremely unreliable I/O. If you're using a fast step-rate and getting frequent "inexplicable" I/O errors, try slowing the step-rate down to 40 ms.

"Delay" refers to the period of time it takes from the moment a drive turns on to the moment it stabilizes at full speed. One second is an adequate allowance for most drives. Some come up to speed even more quickly. Super Utility allows you either of two settings--a full second or a half. As with the stepping-rate, you're safe with a speed that's too slow, but not with one that's too fast. So if you have problems, try slowing down.

The delay is entered and displayed in half seconds. Some older versions of Super Utility allowed 4 settings at quarter second intervals. Some instruction manuals distributed with the current version of Super Utility still state that the delay is indicated in quarters of a second. That is incorrect.

The sample drive zero is set to 2-half seconds (or one full second) delay.

The final item on line one is 'WP=N.' "WP" stands for software write-protect. If you set it to "Y" for "Yes," Super Utility will act as if the disk in that drive had its write protect notch covered, and refuse to write to it.

As stated earlier, the second line of drive data contains information implied by the first line. In our sample, the first item is 'DENSITY=S.' 'S' stands for "Single." The 'A' on the first line specifies a disk containing Model I TRSDOS. That implies single density.

The 'A' on the first line also implies the 'L-GRANS' message on the second line. "L" or "Logical" grans means that the physical, or real, tracks are the same as the relative tracks. If you use double density NEWDOS-80 or DBLDOS, then instead of 'L-GRANS' you will see 'I-GRANS' for "Illogical" grans. This indicates that the physical track numbers do not necessarily agree with the track numbers used by the DOS's track numbering scheme. See the explanation of Disk Relative Sectors (DRS) in the MISCELLANEOUS Chapter for fuller information.

'TRACK0=S' specifies that track zero is in single density. This is meaningful because track zero doesn't always have to be the same density as the rest of the disk. This is especially true for double density Model I disks. The Model I requires track zero, sector zero of any bootable disk to be in single density, even if the rest of the disk is in double density. So some Model I double density operating systems (DOSPLUS, for example,) have a single density track zero. This makes double density DOSPLUS disks bootable on a Model I.

Other Model I double density DOS's (like LDOS) require track zero to be the same density as the other tracks on the disk. This has the advantage of being consistent, and leaving a little more room on double density disks; but if you want to run LDOS with a double density diskette in drive zero, you have to boot in a single density LDOS disk and then swap.

NEWDOS-80 allows you more or less full flexibility. You may have a track zero of either density, regardless of the density of the other tracks. But Super Utility doesn't support all such possible NEWDOS-80 configurations. For discussions of which are supported, see the MISCELLANEOUS Chapter of this book, and note 13 in your Super Utility manual.

At any rate, the diskette in our sample drive zero contains Model I TRSDOS, so track zero, like all the other tracks, is single density.

The next bit of information is 'SS=0.' 'SS' stands for "Starting Sector." In every DOS except Model III TRSDOS, the first sector on each track is zero. Model III TRSDOS doesn't use a sector zero, so the first sector of each of its tracks is one.

Finally, we have 'DAM=0'. DAM stands for Data Address Mark. See the Technical Introduction for more information on DAM's. The DAM's being referred to here are the non-standard DAM's to be found on the disk. Usually, non-standard DAM's are used only for directory sectors. The only known exception is Model III TRSDOS which reverses things. It uses standard DAM's in the directory and Read Protected DAM's everywhere else.

'DAM=0' indicates that the non-standard DAM's are of the Read Protected variety. This is the non-standard DAM used by every DOS other than LDOS. LDOS uses a "User Defined" directory DAM. Super Utility will indicate this with 'DAM=1'.

Now lets take a quick look at the other three drives in our sample configuration table. The two lines for drive one read as follows:

```
+:1B= 80  TKS= 80, DIR= 40, STEP=0, DELAY=1/2, WP=N
          DENSITY=D, L-GRANS, TRACK0=D, SS=1, DAM=0
```

The information on line one indicates that we have, in drive one, a Model III TRSDOS family disk with 80 physical tracks and 80 relative tracks. The directory is on track 40. The drive

read/write head can step from track to track within 6 ms, and the drive comes up to speed within half a second.

Line two describes a double density disk with logical grans. Track zero is in double density; the starting sector of each track is sector one, and the non-standard Data Address Marks are of the Read protected variety.

Now for drive two:

```
=:2C= 35  TKS= 35, DIR= 17, STEP=0, DELAY=1/2, WP=Y
          DENSITY=S  L-GRANS, TRACK0=S, SS=0, DAM=1
```

The "=" at the beginning of line one indicates that drive two is an 80 track drive (narrow track width) containing a diskette which was formatted on a 35 or 40 track drive (full track width). The rest of line one shows that the diskette was formatted with LDOS, has 35 physical tracks and 35 relative tracks. Its directory is on track 17. The drive can complete a track-step within 6 ms and comes up to speed within half a second. It is software write protected.

Line two indicates that the diskette is in single density and has logical grans. Track zero is in single density, the starting sector of each track is zero, and the directory uses the User defined data address mark.

And finally, drive three:

```
-:3I= 35  TKS= 63, DIR= 17, STEP=3, DELAY=2/2, WP=N
          DENSITY=D, I-GRANS, TRACK0=S, SS=0, DAM=0
```

The minus ("-") at the beginning of line one indicates that there is no drive three in the system, so the rest of the drive data is of little consequence. However, if the drive described were logged onto the system, line one indicates that it would be configured to read a NEWDOS-80 diskette with 35 real, or physical, tracks and 63 relative tracks. The directory is shown to be on relative track 17. The step speed of the drive is 40 ms and the drive requires a full second delay.

Line two describes a double density diskette with illogical grans. Track zero is single density. The first sector of each track is sector zero, and the directory has Read protected Data Address Marks.

As I said earlier, some Super Utility users seem to have trouble setting up the configuration table for their own drives. The difficulty nearly always is the result of misunderstanding how the data is entered. Here are three simple rules to follow. They all apply when you're changing the the lines in the configuration table which pertain to the drives:

Rule #1) The first thing you must enter for a drive is either a DOS specifier (a letter from "A" to "J") or "+", "-", or "=".

People often forget this when they want to change the number of physical tracks on a drive. Let's consider an example. Suppose our configuration table shows drive one as follows:

```
+:1B= 40  TKS= 40, DIR= 17, STEP=0, DELAY=1/2, WP=N
          DENSITY=D, L-GRANS, TRACK0=D, SS=1, DAM=0
```

To change the number of tracks from 40 to 35, the first thing to do is to get the modify arrow down to where it's pointing at the "+". So hit <ENTER> until you see the following:

```
=>+:1B= 40  TKS= 40, DIR= 17, STEP=0, DELAY=1/2, WP=N
            DENSITY=D, L-GRANS, TRACK0=D, SS=1, DAM=0
```

Now here's where people have gone wrong. They type "35" and press <ENTER>. Super Utility rejects this input and waits for them to try again. The correct thing to enter is "B35". "+B35" would also work, but the "+" is unnecessary. If you want, you may also put a comma between the "B" and the "35"--e.g. "B,35" or "+B,35". But again, this is unnecessary.

Rule #2) The second line of information for each drive is implied information. You can change it only by changing line one.

Rule #3) Line one has two track-counts. The first is for physical tracks, the second for relative tracks. Line one's second track-count ('TKS= ') is implied, like the data on line two. It can be changed only by altering track-count #1 or the DOS specifier.

Let's go back to the previous example. We want to change

```
+:1B= 40  TKS= 40, DIR= 17, STEP=0, DELAY=1/2, WP=N
```

to

```
+:1B= 35  TKS= 35, DIR= 17, STEP=0, DELAY=1/2, WP=N
```

Here's a WRONG way to enter the information:

```
B35,35
```

People sometimes make the mistake of entering "35" twice, since they want both 40's in the target line to be replaced by 35's. But the first 35 entered will change both 40's to 35's. If you enter "35" twice, Super Utility will think you want it to use the second 35 for the drive's directory track.

The correct way to enter the change is:

B35 or B,35.

Here's an input line that will change every item on line one in this example.:

=I35,20,3,2,Y

The "=" sets the drive to double-step. The I specifies a double density NEWDOS-80 operating system with a single density track zero. The "35" specifies a 35 track diskette. Since NEWDOS-80 uses Disk Relative Sectors, the combination of "I" with "35" sets the relative track-count to 63. The 20 tells Super Utility that the directory is on track 20. Since this is a DOS that uses data relative sectors, the "20" indicates relative track 20, not physical track 20. The "3" sets the drive to a 40 ms stepping speed. The "2" sets the drive for 2 half-seconds (one full second) delay. And the "Y" turns on software write-protect for drive one.

The new information in the configuration table will look like this:

```
=:1I=35,  Tks= 63, Dir= 20, Step=3, Delay=2/2,  WP=Y
          Density=D, I-Grans, Track0=S, SS=0,  DAM=0
```

Notice that line two has also changed. It now indicates a diskette with illogical grans, a starting sector of zero, and a single density track 0.

Using the configuration module to reconfigure Super Utility is called "soft configuring." Every time you boot Super Utility the original configuration will be restored and you will have to reconfigure Super Utility for your system. You may prefer to have Super Utility boot up preconfigured to your specifications. You can accomplish this by zapping certain sectors of your Super Utility disk. This is called "hard configuring."

Hard configuring also lets you change certain aspects of Super Utility that aren't alterable through soft configuring. You can change the default disk name and date that Super Utility uses during formatting and Repair GAT. You can also change the formatting fill pattern. "Fill pattern" refers to the bytes put into "blank" areas of a disk during formatting. Also, if your computer has been modified, hard configuration may allow you to initialize some of the special features of your computer. More on this shortly. You may also configure the boots Super Utility uses for "Repair Boot," by choosing boots that are compatible with your favorite DOS's.

Hard configuring is very easy, but you should be thoroughly familiar with Super Utility's zap program before you attempt it. If you aren't yet familiar with Zap, back up a disk and use Zap experimentally on it. When you have tried most of Zap's features and feel at home in the modify mode, you are ready to hard-configure your Super Utility diskette.

The hard-configuration directions in the Super Utility manual are quite clear and there is no need to rehash them here. However, there are a few things that should be pointed out. Hard-configuration lets you have Super Utility send out any bytes you chose to any ports you choose (up to a maximum of twelve port outputs). Port addressing is used by a variety of special TRS-80 add-on hardware, such as the Methuselah real time clock, RS232 boards, high speed mods, and reverse video boards. If you have any such devices, you may find that they require software-initializing to work with Super Utility.

This is arranged for by hard-configuring--specifically by zapping sector five of track one (for Model I's) or track two (for model III's). In general, you will use sector five hard-configuring to have Super Utility initialize your port addressed hardware. There is, however, one exception. The initialization of devices which are controlled by port 254 is best hard-configured in another manner.

Super Utility uses port 254 in conjunction with its clock speed control. Therefore, any port 254 initialization which you do via sector five hard-configuration may be countered by Super Utility when it initializes the clock. Therefore, it's best to use the clock 'ON=...' and clock 'OFF=...' to initialize your port 254 addressed special hardware.

I'll consider two examples:

1) You have a high speed mod and a reverse video mod, each of which uses port 254.
2) You no high speed mod, but you do have a reverse video mod which uses port 254.

In example #1, you have a high speed mod. The least significant bit of any value output to port 254 controls the speed. In other words, OUT 254,1 turns on high speed and OUT 254,0 restores normal speed. This is a fairly standard way of doing things.

Now, suppose you also have a reverse video mod which is controlled by the second least significant bit of any value output via port 254. In other words, OUT 254,2 would turn on reverse video, and OUT 254,0 switches back to the normal display mode.

As a final supposition, let's assume that you would like Super Utility to come up in both reverse video and high speed. To turn both on simultaneously, a value with both the least significant and second least significant bits set must be sent to port 254. OUT 254,3 would do the trick.

All you have to do is hard-configure sector one to

'HighSpeed =Y' and 'ON =3E03D3FE0000'.

This disassembles as:

```
LD   A,03
OUT  254,A
NOP
NOP
```

The net effect is to send a "3" to port 254.

It would probably also be a good idea to configure 'OFF=3E02D3FE0000'. This disassembles as

```
LD   A,02
OUT  254,A
NOP
NOP
```

The net effect of this code is to send a two to port 254. The advantage, in this case, of turning off the high speed clock with a two rather than a zero, is that the two won't turn off the reverse video as well, whereas a zero would.

Example #2 assumes that you have no highspeed mod, but you do have a reverse video mod which is turned on and off with the instructions OUT 254,1 and OUT 254,0. You want Super Utility to turn your reverse video on, and keep it turned on.

The easiest thing to do is to use hard configuration to tell Super Utility that you don't have a high speed mod. Also tell Super Utility that 'OFF=3E01D3FE0000,' or in effect, OUT 254,1. This will keep your reverse video turned on.

A number of Super Utility users have been confused about hard-configuring the boots. The hard-configured boots are used when you ask Super Utility to repair a disk's boot sector. At that time, Super Utility simply writes one of its hard-configured boots over the damaged boot on your disk. Since no one boot is compatible with every DOS, you may need to replace the repair-boots that came on your Super Utility disk with others.

Super Utility comes hard-configured with four separate boots, but only two are available for use at any one time. Remember, your copy of Super Utility will load and run on both the Model I and Model III. You have one pair of boots available when working on the Model I and another pair when working on the Model III. Each pair consists of a Model I boot and a Model III boot. That way, no matter which machine you're working on at any given time, you may repair disks belonging to either model.

Kim felt that if you work on both models, you may have a different set of disks containing different operating systems with each model. Therefore, you might need a separate set of repair-boots for each.

Whenever you use Super Utility's Repair BOOT Sector option, Super Utility actually writes both of the current pair of boots to the target disk. The Model I boot gets put on track zero, sector zero, where the Model I would expect to find it. The Model III boot is placed on track zero, sector one, where the Model III expects to find its boot. The one exception to this occurs when you repair a Model III TRSDOS disk. Since the target disk doesn't have a sector zero, the Model I boot doesn't get written.

To ensure that Super Utility will use a boot that is compatible with your DOS, simply use Zap's Copy Sectors routine to copy the boot sector from the DOS onto the appropriate Super Utility hard configuration sector.

The boot sector of your DOS disk is track zero, sector one, for Model III TRSDOS disks, and track zero, sector zero, for all others. The appropriate Super Utility hard configuration sectors are identified in you Super Utility manual, and by prompts which you will see as you Zap through the Super Utility hard configuration sectors.

Chapter III
WHAT TO DO WITH A MYSTERY DISK

When Super Utility won't behave, the reason is almost always the same. The program is not configured correctly for the disk(s) being worked on. If you don't understand Super Utility's configuration module, the previous chapter of this section of this book zooms in on the topic. Also read Kim's AFTERWORD, included in this book.

If you understand the configuration process, you may still feel caught up in a kind of Catch-22. Super Utility won't read your disk unless you first configure correctly. You can't configure until you know some facts about the disk--such as the computer it's supposed to run on (Model I or III), what DOS it contains, whether it's single or double density, how many tracks it has, where the directory is located, and whether it was created on a 40 or 80 track drive. You can't get this information until Super Utility lets you look at the disk. So where do you start?

If you have no idea of what a disk contains, density-wise, DOS-wise, and every other -wise, use Zap's Read ID Address Marks command (option "A" from the Zap menu). This can give you a good deal of information about the disk. Here's the procedure.

Go to the configuration module. Make sure 'SAVE=N'. Then reconfigure the drive you're going to use as follows: configure the drive to be in the system. Do not set it to double-step, and make sure it's configured to the highest possible number of tracks the drive is physically capable of supporting. For instance, suppose you're going to use drive one and it's an 80 track drive. You look at the configuration table and see that SAVE='Y' is currently in effect. Type N,N,N (assuming you don't care for high speed or DUAL). Then press <ENTER> until get the "=>" gets down to the line for drive one. Then enter

+A,80

In this case, the "A" is a dummy parameter, it doesn't mean we think the DOS is TRSDOS. You could just as well enter any other DOS specifier.

Make sure the configuration table has been updated correctly. Then press <SHIFT><BREAK> to return to the main menu. Press <ENTER> to go to to the Zap utility. Then enter "A" to select the "Read ID Address Marks" function. Finally, enter "1" to choose drive one. Super Utility will now try to read the DAM's for drive one, track zero.

There are two major possibilities:

Possibility #1. Super Utility completely or partially succeeds at reading track zero's DAM's. If it succeeds completely, the screen will be filled with a scrolling display under all the headings except the last three (if there is information under the last three headings as well, hold down the <X> key until that data disappears--see note 1).

If Super Utility has partial success at reading track zero's ID address marks, you will see information scrolling up the screen, interspersed with error messages.

Possibility #2. Super Utility is completely unable to read track zero's ID address marks. Error messages appear on every line of the display.

If Super Utility can read track zero, make note of its density. The density is indicated by either an "S" for "Single," or a "D" for "Double," in the the column under the "u" of the word "Source."

If Super Utility can't read track zero, it could mean any of several things. Assuming your system is in good repair, you might be trying to read an unformatted disk. If your system doesn't have a double density board, you might be trying to read a disk whose track zero was formatted in double density. Or you might be trying to read a badly farkled (or messed up) disk.

Hold down the up-arrow key. Super Utility will attempt to read subsequent tracks. If it succeeds with all tracks other than zero, and your system doesn't have double density, you may be trying to read a NEWDOS-80 disk which was created with a double density track zero, but with all the other tracks formatted in single density. However, it's unlikely anyone would format a disk in such a manner. What is more probable is that something farkled track zero without affecting the rest of the disk.

If Super Utility succeeds at reading track zero, there are two special "wrong drive-type" patterns to be alert for. In one, the track number shown under the heading "Track" increases by twos, while the track count under the heading "Source" increases by ones. This probably means you're using a 35 or 40 track drive to read a disk which was formatted in an 80 track drive. Try switching the diskette to an 80 track drive and starting again.

In the other "wrong drive-type" pattern, the track count in the "Source" column goes up by twos, while the track count in the "Track" column goes up by ones. Also, every other time the head steps (you should hear a tick from your disk drive each time it steps), you will get a series of "ID Read Error" messages. This indicates that you're using an 80 track drive to read a diskette which was formatted in a 35 or 40 track drive. Either switch the disk to a suitable drive, or go back to Super

Utility's configuration module and configure the drive you're using to double-step (use an equal sign ("=")).

Once you have Read ID Address Marks happily reading your disk, hold down the <X> key until Super Utility starts filling in the last three columns of information (see note 1). Keep your eye on the "Data" column. This tells you the DAM type for each sector on the track. Now press the up-arrow repeatedly to step through the tracks (See note 2).

With any DOS other than Model III TRSDOS, the "Data" entry for all non-directory sectors should say 'S>td,' for "standard." When the entries in the data column change from S>td to 'R>ptc' (for "read protected") or 'U>df' (for "user defined"), you have located the directory track. With TRSDOS III, things are switched around. The directory has "standard" DAM's and the rest of the disk has "read protected" DAM's.

The directory will usually be on track 17 decimal for 35 or 40 track diskettes. For 80 track disks, the directory is normally on or near track 40. If the DAM was "U>df," you're looking at an LDOS disk. "R>ptc" indicates any operating system other than LDOS.

While looking at the directory track, use the space bar to freeze the action. Look for the lowest numbered sector. If you don't see a zero, start the scrolling again by pressing <ENTER>, and freeze it with <SPACE> again. Repeat this procedure several times, each time checking for the lowest numbered sector visible on the display (you might not catch the track's lowest sector on the screen the first few tries). If the lowest sector you can find is one, you're probably looking at a Model III TRSDOS diskette. All other standard DOS's should start each track with sector zero.

Another thing to watch for is whether the directory, as revealed by the change in DAM's, starts at the beginning of its track, or somewhere in the middle. If the directory doesn't occupy the whole track, then you'll see some sectors with S>td DAM's, and other sectors in the same track with R>ptc DAM's. In that case, the directory will probably "wrap around" and occupy some of the next track as well. When this occurs, you may be fairly certain you're dealing with NEWDOS-80 or DBLDOS.

Take note of the number of the directory track(s), and its density. Then, press <X> to go back to mode one (see note one). Hold down the up-arrow and let auto repeat take over. The read write head should step in track by track.

When you reach a track whose ID marks can't be read, Super Utility will "stick." If you get an error message that says "Not Formatted," or "ID Read Error," and Super Utility doesn't find any sectors for that track, you may have past the last formatted track. If the source track is 35, 40, or 80, this is especially likely. To be certain, you may try forcing the head to search for more tracks further toward the center of the disk. To do this, press the up-arrow a few times.

When you have found and noted down the last track on the disk, there are a couple of other places on the disk to inspect--the Boot and GAT sectors. Start by pressing <SHIFT><BREAK> to get back to the main menu. Then go to the configuration program. Configure the drive you're using to the correct density, track number, and directory track number your inspection has revealed.

Here's how to configure for the DOS. First of all, you may have already discovered which DOS the disk contains. If the directory had U>df DAM's, then the DOS is LDOS. Use "C" if the tracks were in single density, and "D" if they were double. If there were no sector zeros, then the DOS is Model III TRSDOS. Use "B."

If the directory started in the middle of a track and wrapped around to the next track, the DOS is NEWDOS-80 or DBLDOS. If track zero is single density and the rest of the disk is double, you won't be able to determine which of the two it is until you examine the GAT. Use "F" in the mean time. If track zero is double density, NEWDOS-80 is indicated. Use "J."

If you don't know the DOS, use "A" for any all single density disk, "D" for an all-double density disk, or "F" for a double density disk with a single density track zero.

Go to Zap's Display Sector program. When prompted with "Drive, Track, Sector", input the target disk's drive number, and default on the other two values. Super Utility will display track zero, sector zero or one, depending upon whether you're looking at Model III TRSDOS or some other DOS.

If the sector you see is mostly zeros, your looking at a data (or non-system, or non-booting) disk. If the sector is full of data, you may be looking at a booting disk, or it might still be a data disk. You should also see an all too familiar message or two near the end of the sector--"DISK ERROR" or "NO SYSTEM."

While you're looking at the boot sector, look at the third byte. For most DOS's, that should contain the number of the directory track. If it doesn't agree with the number you came up with when you used Read ID Marks, it could mean one of two things.

1) You may be confused because Read ID Marks gave you the track number in decimal, but the byte in the boot sector in in hex. Convert the number base of one of the values and see if they agree after all.

2) You may be looking at a DBLDOS, NEWDOS-80, or TRSDOS III disk. These are the only DOS's whose third byte on the boot sector doesn't have to point at the directory. DBLDOS has no directory pointer at all. TRSDOS III uses the second boot-byte, rather than the third, as the directory pointer.

Double density NEWDOS-80 disks with single density track zeros maintain a second boot sector at relative track zero, sector zero. This happens to be the same as physical track one, sector zero. It is this second boot whose third byte is required to point to the directory track.

3) You may be looking at a NEWDOS-80 disk. Boot byte three is pointing at the directory, all right. But since it uses "illogical" tracks, it won't seem to be pointing where we would expect.

You might think that if the answer were (3), we'd have already known that we were looking at NEWDOS-80. After all, when we ran Read ID Marks, we'd have discovered the directory to be spread across two tracks. However, this particular NEWDOS-80 disk might just happen to have its directory contained within a single physical track despite its "illogical grans". If such were the case, we wouldn't yet know what we were dealing with.

The next thing to do is use Zap to look at the GAT sector--the first sector in the directory. The GAT sector contains the disk name, date, the hash of the master password (which will probably be unrecognizable), and the AUTO command (if any). There is a good chance that the disk name will identify the DOS for us. This is especially true if the disk is a system, rather than data diskette.

If you go through the whole rigmarole discribed above and still don't discover what kind of disk you're looking at, try zapping through the disk. System modules usually have ASCII copyright messages embedded in them.

When you know the disk's DOS, density, track-count, etc, go to the configuration module and update the configuration table to incorporate all the hot new information. Finally, run Zap's verify program and Disk Repair's Check Directory program. While you're in the Disk Repair menu, use option eight to display the diskette's directory. If it has lots of "/SYS" files ("/DOS" files on Multidos), you're probably looking at a system disk. If only BOOT/SYS and DIR/SYS are listed, you've got a data disk. By now you should have a comprehensive picture of the mystery disk.

Chapter IV
DISK ERRORS

Before getting too deeply into errors and error recovery, there is one point I would like to emphasize. Super Utility's features far exceed the capacities of competing programs, but it can't do miracles. So don't get overconfident. The most important step you can take to avoid the loss of crucial data is preventive. Please make backups.

Another worthwhile prophylactic measure is to periodically perform a Verify Sectors and Format without Erase on all important disks. In other words, pretend a problem exists and follow the problem recipe outlined below. This may seem a lengthy process, but it is a time-investment which, in the long run, may save you many more hours than you put in.

One simple way to classify disk errors is to break them up into two categories--minor and drastic. Minor errors are those in which only a small number of bytes have been clobbered. Drastic errors indicate massive alteration or obliteration of disk data. Super Utility can correct most minor disk errors. But when drastic errors occur, you'd better have those backups handy.

The techniques I'll be describing here deal with the repair of errors which fall into the "minor" category. When a minor error occurs, it usually restricts itself to one of the two subfields of one sector. I will refer to errors in the header subfield as header errors, and errors in the data subfield as data errors.

Another way to lump errors into categories is to separate them into errors which occur in the directory and errors which occur elsewhere on the disk. Directory errors may be much more critical than others, or they may be much simpler to deal with, depending on whether or not they're in one of the first two directory sectors.

The first two directory sectors contain the Granule Allocation Table (GAT) and Hash Index Table (HIT), respectively. The unique thing about these tables is that the crucial information they contain may be derived from the other directory sectors. Therefore, if one or both of these sectors is wiped out, Super Utility can reconstruct them for you, almost completely automatically--as long as the rest of the directory is intact.

Now that we've had a quick overview of the taxonomy of disk errors, let's take a closer look at header errors. One insidious thing about them is that, though they are minor in the sense of only involving a byte or two, they are still drastic in the sense of being nearly impossible to fix. A sector which contains a bad header may be made usable, by Format without Erase. But the data in the bad sector might be beyond recovery.

Remember, the header contains four bytes of information which identify and define a sector--namely the track number, head number, sector number and sector length. These bytes, you will recall, are preceded by a one byte ID mark, namely FE hex,

and followed by a two byte CRC. Any alteration to the data in this area makes it impossible for DOS or other software to access the sector in question; usually the afflicted sector can't even be located!

When a sector becomes totally lost to DOS, trying to access it results in error messages like: "Sector NOT FOUND" (this is Super Utility's own error message), "SEEK ERROR DURING READ" (TRSDOS error 02), or "SEEK ERROR DURING WRITE" (TRSDOS error 10). The message you get may vary, depending on the DOS (or other software) attempting the disk I/O.
NOTE: All the DOS error numbers in this section are in decimal.

If a header CRC error occurs, DOS may be able to find the sector, but the data it contains will still be inaccessible. This is because the FDC generates an interrupt when it detects the error condition, and doesn't read any further. In such cases, you may get a message such as "Sector NOT FOUND, ID CRC Error" (Super Utility's message), "PARITY ERROR DURING HEADER READ" (TRSDOS error 01), or "PARITY ERROR DURING HEADER WRITE" (TRSDOS error 09).

One thing about these errors you should be aware of is that, despite the error messages, they are not parity errors, in the normal sense, but CRC errors. "Parity" usually refers to a one bit flag which indicates an odd or even number of one-bits in a given byte. The CRC is a 16 bit indicator which reflects a great deal more than a one-bit count.

If you get a header (or ID) error, it's time to go for the backup. If you don't have a backup, and are really desperate to recover the data, you may be able to do so, at least partially. The method involves doing a track read, printing out the result, reformating (without erase) the track with the problem sector, and re-entering the lost data by hand, using the Modify Mode of Zap's Display Sector module. This is a somewhat laborious procedure, and it often doesn't succeed. However, it's a worthwhile last ditch attempt to recover from a variety of wipeouts. I describe the procedure in detail under the heading TRACK RESCUE.

Errors which affect a sector's data subfield may be much easier to deal with. If the Data Address Mark got clobbered, you're out of luck--it's like an error in the header subfield; the sector may be totally lost. DOS will tell you something like "DATA RECORD NOT FOUND DURING READ" (TRSDOS error 05) or "DATA RECORD NOT FOUND DURING WRITE" (TRSDOS Error 13). Super Utility will probably give you the good old "SECTOR NOT FOUND" error message. The exception to this is an error in which the DAM gets changed into one of the other legal DAM bytes (there are four possible DAM's for the Model I, and 2 for the Model III). In such a case, Super Utility will be able to restore the disk.

By the way, don't confuse the "DATA RECORD NOT FOUND DURING READ" type message with one like "LOST DATA DURING READ" or "LOST DATA DURING WRITE" (TRSDOS errors 03 and 11, respectively). These are two entirely different kettles of error. "LOST DATA" doesn't necessarily indicate anything wrong with the diskette at all. It means that the CPU couldn't catch the data as fast as the FDC was grabbing it from the disk. The problem is often that the drive is spinning too fast. Have it timed!

Except for clobbered DAM's, a data subfield error may be fixed relatively easily. Super Utility's message, when it encounters such a sector, is "DATA CRC Error." DOS may present you with a message like "PARITY ERROR DURING READ" (TRSDOS error 04) or "PARITY ERROR DURING WRITE" (TRSDOS error 12).

Here's a recipe for dealing with problem disks:

PROBLEM RECIPE

1) Follow the instructions under the heading "VERIFYING A DISK". This will show you how to test the disk with Super Utility's "Verify Sectors" program. Then come back to this recipe.

A) If the final result of "Verify Sectors" is "0 bad sectors," even though you may have needed several retries on some sectors, follow the instructions under the heading "FORMAT WITHOUT ERASE".

B) If the final result of "Verify Sectors" is more than zero bad sectors, then follow the instructions under the heading "STUBBORN ERRORS."

2) Follow the instructions under the heading FIX-DIR.

END OF PROBLEM RECIPE

One other error message which DOS may occasionally give you is "ATTEMPTED TO READ SYSTEM DATA RECORD" (TRSDOS error messages 06 and 07). According to the TRS manual, this is usually the fault of a user program. What is actually happening is that DOS is trying to read a sector which has incorrect DAM's. This could happen under a number of circumstances. One possible cause could be your using Super Utility's Read Protect Directory program and give the wrong address or length for the directory. The careless use of Zap's Alter Data Address Marks could have the same effect.

To overcome the problem, use Zap's Read ID Address Marks to locate all sectors with incorrect DAM's. Remember, except with Model III TRSDOS, all non-directory sectors should have

"standard" DAM's. Model III TRSDOS should have "standard" DAM's in the directory and "read protected" DAM's everywhere else. Use the Alter Data Address Marks option to correct aberrant sectors.

VERIFYING A DISK

If your disk seems to have errors, boot up Super Utility, go into Zap, and use the Verify Sectors program (Zap selection #2) to go over the problem disk. If Super Utility isn't configured correctly for the target disk, be sure to use the proper DOS and track-count overrides with the Verify command.

Each time Verify Sectors reports an error, make a note of the track and sector number displayed, and the type of error. If you have a printer, instead of taking notes, you can set 'DUAL=Y' from the configuration module, before starting this procedure. Retry reading each problem sector several times (if you're desperate, use the <C>ontinuous or <N>onstop selections and retry every problem sector for a couple of minutes).

At this point I would like to establish a couple of definitions:

1) From now on, I will refer to problem sectors which Super Utility succeeds at reading after some retries as compliant problem sectors.

2) I will refer to sectors which Super Utility can not read, even after a large number of retries, as stubborn problem sectors.

As you note down each problem sector, also record whether it is compliant or stubborn. If a problem is stubborn enough to defy four or five read attempts, you can usually write it off as completely stubborn.

Compliant problem sectors may be repaired automatically by the Format without Erase routine. But Format without Erase will be one of the last steps in fixing your disk, because a premature Format without Erase can make it impossible to fix certain stubborn problems which other techniques might save.

When Verify Sectors is finished, it will display the number of stubborn sectors (it will call them bad sectors). Take note of this number, and return to the Problem recipe.

FORMAT WITHOUT ERASE

WARNING: It is extremely important to correctly configure Super Utility for your disk before using Format without Erase. Either use Super Utility's configuration module or use the proper DOS specifier with each Format without Erase command. If you use DOS specifiers with Format without Erase, you should use a track-count override as well. If you don't know how to enter DOS specifiers and track-count overrides, the section on disk-extending in the MISCELLANEOUS Chapter contains an example

which should be helpful.

Format without Erase is item number three on the Format menu. When you run it, it will pause at the first problem sector it finds (see note 1). If your notes indicate that this is a stubborn sector, just use <S> to skip it. If, on the other hand, it's a compliant one, enter <C> and let Super Utility retry the sector until it is read and reformatted--or, if your notes indicate that the current problem sector is the first of several compliant problem sectors, with no intervening stubborn ones, enter <N> and let Super Utility zip through them all automatically.

After Format without Erase has gone over the entire disk, it will go back to track zero an verify all it has done. Unless your disk is physically damaged or worn, this verify should not encounter any problems. Finally, Format without Erase will update the directory.

The directory update does two things. For one thing, it can increase the number of available tracks in the allocation table. This is unlikely, but it can happen as a result of your "extending" a disk the hard way. To extend a disk is to increase the number of tracks it has, without losing any any of the data currently on the disk. E.g, you may extend a 35 track disk into a 40 track disk.

The normal way to extend a disk is to use Super Utility's Standard Format routine. This is a quick and highly automatic method. I describe it in this book in the Miscellaneous Chapter. Another way to extend diskettes is via the Format without Erase routine.

Suppose you want turn a 35 track diskette into a 40 tracker using Format without Erase. You'd go to the Format menu and select the Format without Erase option for the disk in question. When you specified the drive number, you'd use an override command to indicate the new number of tracks--e.g, if the target disk were on drive zero, you'd answer the 'Drive(s) ?' prompt with

0=40

When Format without Erase started the 36th track (track number 35, since the first track is numbered zero), you'd get an I/O error, because that track was as yet unformatted. You'd enter <S> to skip the sector, and receive a similar message for the next sector. To add five tracks to a single density diskette (at ten sectors per track), you'd have to skip 50 sectors. If you were extending a 35 track double density diskette (18 sectors per track) to 40 tracks, you'd have to enter <S> 18*5 times, for a grand total of 90 S>kips.

This is certainly an unnecessarily laborious way to approach the task of extending a disk--especially when you could achieve the same end so easily with the alternate technique I

describe in the MISCELLANEOUS Chapter. But if you were to follow the above procedure, the Format without Erase routine would faithfully update the directory to the new track-count, as soon as it verified the reformatting process.

The second thing Format without Erase does when it updates the directory is to liberate any tracks which which were previously locked out. This means that you'll have more free space available. But if a gran had been locked out because the disk was flawed, data later saved to that gran may be in jeopardy. So if you rescue a questionable disk with Format without Erase, back it up a couple of times and throw away the original. Or keep the original for a scrap disk--but don't entrust important data to a disk which may be physically flawed.

After updating the directory, Format without Erase will tell you how many sectors it couldn't read on its first (reformat) pass, and how many it couldn't read on its second (verify) pass. Remember, the data in any sector which failed on the first part has been lost, even if it was readable on the second pass.

This is the end of the Format W/O Erase section. Return to the Problem Recipe.

STUBBORN ERRORS

There are two kinds of stubborn errors: 1) "peccadilloes", and 2) "nightmares" (sometimes there's just no getting around technical language). I'll consider each in turn.

1) Peccadilloes

Peccadilloes are errors which result in the the Super Utility error message: 'DATA CRC Error'. Remember, every sector has a data subfield, and a two byte CRC is recorded at the end of that field each time data is written to the sector. "DATA CRC Error" means that there's a discrepancy between the CRC recorded when the data was written, and the CRC which the FDC computes when it reads the data at a later time. This indicates that something in the data subfield has changed since the sector was last written to.

There are two ways such a change could happen. One way, which I'll call a data peccadillo, occurs when one or more bytes of actual user data gets altered. The other way, which I'll call a CRC peccadillo, occurs when one or both of the recorded CRC bytes to get corrupted, while the user data remains intact. Either type of peccadillo results in disagreement between the recorded CRC and any freshly computed CRC.

Both kinds of peccadilloes can be fixed just by reading and rewriting the problem sector with Zap's Display Sectors option. Ask Zap to display the problem sector. When you get the "DATA CRC Error" message, just select the mini-menu's <S> option. Super Utility will then display the data as it found it.

Next, you enter the modify mode by pressing <M>. The last step is to key <ENTER> <U> <ENTER>. (Editor's note: <ENTER> <ENTER> would work just as well.) This resaves the sector. When the sector is resaved, the FDC automatically recomputes new checksums, which are in accord with the rest of the data, and saves them on the disk in place of the old ones.

If the peccadillo in question was a CRC peccadillo, all this is fine. But if the error was a DATA peccadillo, the data in the "fixed" sector will still be corrupt, though Super Utility, DOS, and other software will now be able to read the sector without generating an error condition.

Unfortunately, both types of peccadillo look the same to the FDC. Only the user has a chance of telling which kind took place. The best thing to do is to inspect the sector while you're still looking at it with Zap. If you're looking at a word processing file, editor/assembler source file, or other ASCII text, it should be fairly easy to spot corrupt bytes. On the ASCII side of the display, you'll see graphics, or other meaningless characters, where there should be legible material. Just enter the modify mode (making sure to set data-entry to ASCII), and type over the garbage.

A word of warning is in order. Sometimes a word processor's text formatting characters, or other information, may look like garbage. If you have Lazy Writer files with underlined sequences, those sequences will look like trash to any disk monitor, including Super Utility. Editor/assemblers often have line numbers or file headers with high bits set. Again, these can look like garbage when viewing disk sectors. I haven't yet seen Mumford's Instant Assembler, Disk Version, but I understand it uses a space saving tokenized source format, both in memory and on disk. Such files will probably also look trashy when examined with Super Utility. So always try to know what you're doing before zapping a disk. If possible, look at the corresponding sector in a backup file, before making any changes.

If the file is not a text file, than it's much more difficult to decide whether your peccadillo was CRC or data, and if data, which data. If you're looking at a BASIC program, in its normal compressed format, than you would have to hand detokenize it, and sort out all the line numbers, and pointers to line numbers, to get an idea of what belongs and what doesn't. This book is not the place to describe such procedures.

If you're looking at machine language object dump, things can get even more difficult. You would have to disassemble the code, and see if anything looks screwy. In doing so, you would also have to sort out DOS's load file format codes. If you don't know what load file format codes are, there's some information on them in the 'PATCH SECTORS' section of this book. If you have no knowledge of machine language, then this book can't help you decipher farkled object files. But if you have a backup of the

file, you can just follow the formula described below.

Assuming you have a backup, it may be easiest to simply copy the backup file onto the problem file. However, sometimes this may be undesirable. If you've updated the problem file since the last backup was made, copying the entire backup file would wipe out the update changes. But copying only the afflicted sector would preserve the update work, providing that the target sector was not one of those affected by the update.

I've noticed that every time I mention something in this section, I seem to have to point out that there are two kinds of that thing. Now I must say that, for the purposes of this discussion, there are two ways a file may be backed up. One is by backing up the entire disk which contains the file. For some perverse reason, a disk which is a backup of an identical disk is called a "mirror image" backup, even though it's not a mirror image, but a true replication.

The other way to backup a file is to just copy it over to a different disk, or a differently named file on the same disk. I'll call this type of backup a "file-backup."

In the case of a mirror image backup, the backup sector will be in the same relative position on its disk as the problem sector is on its own. But when you work from a file backup, the problem sector and its backup sector will probably occupy different places on their respective disks (especially if both the problem file and its backup are on the same disk). But the problem sector will still have the same relative position within each file. In other words, the target sector should have the same File Relative Sector Number (FRSN) in both the problem file and backup.

When copying a problem file from a mirror image backup, you can use Zap's Copy Sectors option. Just specify the same track and sector for both the source and destination, varying only the drive number if you're using more than one drive.

If you're working with a file backup, there are two possible procedures which are useful for finding the proper backup sector. Which is better? That depends on the individual situation and your temperment.

METHOD A

First, determine the FRSN of the problem sector. To do this, go to Super Utility's File Utilities section and run option one, Display File Sectors. Enter the name of the problem file. If the disk containing the backup file is also on line at the time, be sure to include the correct drive number as part of the file spec.

Super Utility will find the file and display various facts about it, and then prompt you with 'Choice ?'. At that time, it's waiting for you to enter the FRSN of the sector you want displayed. Take a guess at the proper FRSN or just press <ENTER>

to default to zero (remember, the number of the first item in a relative count is usually zero). Use the arrow keys to step through the file until you come to the problem sector. You can identify it by its track and sector number, which are displayed on the left side of the screen, just as they are in the ordinary Zap mode.

Note the problem sector's FRSN. This will be displayed near the screen's lower left hand corner, to the left of the hex digits "E0"--under the heading 'RSEC', for Relative Sector. If the problem prevents Zap from loading the sector, you can still determine its FRSN by the FRSN's of the adjacent sectors.

Next, use Display File Sectors to look at the backup file. When Super Utility has located the file and asks you to enter your choice of sector, enter the FRSN which you have just determined on the original. The sector displayed should be the backup of the problem sector. Make a note of its track and sector number as displayed at the left side of the screen, next to the hex digits "70" and "90", respectively.

You can now use Zap's Copy Sectors option to copy the backup sector onto the problem sector.

METHOD B

Use File Utilities' Compare Files to compare the problem file with its backup. If all goes well, Super Utility will report data differences in only one sector, and that should be the problem sector.

Super Utility will report the differing sector(s) in terms of its FRSN. Take note of this number and go to Super Utility's' File Utilities section. Use Display File Sectors to view the problem file. When you see the 'Choice ?' prompt, enter the FRSN displayed by Compare Files. Super Utility will try to read and display the sector. If it succeeds, the track and sector numbers will be displayed in the usual Zap format. If it fails, they will be displayed in the error message. In either case, note them down. Then do the same with the backup file. Finally, copy the backup sector onto the problem sector.

2) Nightmares

An ancient computer blessing goes, "May your nightmares all have backups." This is especially true when a nightmare error occurs on a directory track (except for the HIT and GAT sectors. If that should happen, only a mirror image backup of the disk can restore things--unless you want to try to hand reconstruct the blown directory sector.

There is a major decision you must make in dealing with nightmares--you must choose and follow one of two divergent paths. 1) Format without Erase, and 2) Track read reconstruct.

If you have a backup, the decision is easy. Format without Erase, and then recopy the lost sector(s). But remember, Format without Erase does destroy the data in nightmare sectors--only compliant errors are cured.

If you don't have a backup, you may still opt for path 1. You'll lose some data, but at least you'll be able to reuse the disk (though I would consider a nightmare prone disk a poor choice for most projects).

If the nightmare occured in a directory sector (other than the GAT or HIT), you'll loose a great deal more than a single sector's worth of data. Any file whose primary directory entry was in the damaged sector will become inaccessible. Files whose extended directory entries were contained in that sector will probably become useless as well.

Since a directory sector can hold up to eight entries, you may loose eight files in one fell swoop. If one or two of those files are crucial system files, the entire disk may become inaccessible to you--especially if you have a one-drive system.

If you're willing to sacrifice the other files cataloged in that directory sector, the problem of the lost system files is one of the easiest to cope with. Most DOS's have their system files in the same place on every disk. The placement of the system files' directory entries is also fairly constant. So you can Format without Erase the nightmare disk. Then copy the corresponding directory sector from another disk onto the nightmare sector. Just make sure that the source disk has the same DOS as the nightmare disk, is in the same density, and has the directory on the same track.

Finally, you might want to use Zap's modify mode to zero out those sections of the directory that don't apply to the system files. Non-system directory entries are probably invalid for the disk you copied them to. If you prefer to experiment, you may leave the entries in the directory and try to access the files they describe. If the files don't work, then kill the the files, or go back and zero their directory entries.

In case you're not at all familiar with directory structure, Figure 5 depicts a typical directory sector. It contains eight entries, each of which takes up two lines of the display. If you look at the ASCII side of the display, you can make out the file names embedded in the first line of each entry.

The file name extensions are all the way to the right. Those files whose extensions are SYS are system files (Multidos uses "DOS" instead of "SYS"). So if a line ends with SYS (or DOS), leave it and the following line alone when you zero out the non-system entries.

Note: A TRSDOS III directory does not code the system files in the "normal" manner, so you will not see any entries for system files in the directory of a TRSDOS III disk. If you must attempt to recover or reconstruct a farkled TRSDOS III

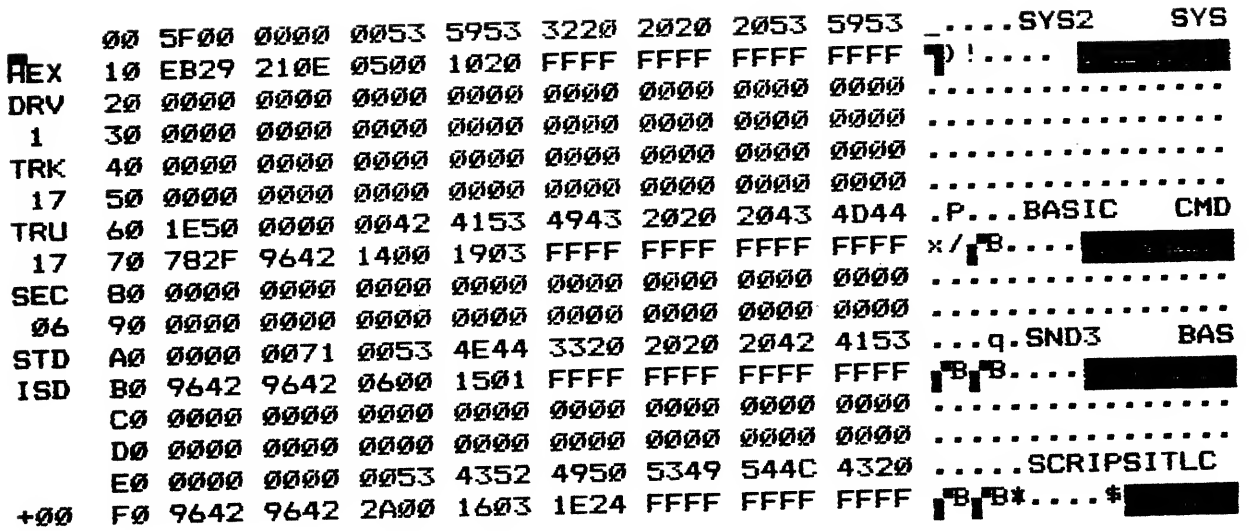

ØØ 5FØØ ØØØØ Ø053 5953 322Ø 2Ø2Ø 2Ø53 5953 _....SYS2 SYS
REX 1Ø EB29 21ØE Ø5ØØ 1Ø2Ø FFFF FFFF FFFF FFFF) !....
DRV 2Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
1 3Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
TRK 4Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
17 5Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
TRU 6Ø 1E5Ø ØØØØ Ø042 4153 4943 2Ø2Ø 2Ø43 4D44 .P...BASIC CMD
17 7Ø 782F 9642 14ØØ 19Ø3 FFFF FFFF FFFF FFFF x/ B....
SEC 8Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
Ø6 9Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
STD AØ ØØØØ ØØ71 Ø053 4E44 3320 2Ø2Ø 2Ø42 4153 ...q.SND3 BAS
ISD BØ 9642 9642 Ø6ØØ 15Ø1 FFFF FFFF FFFF FFFF B B....
CØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
DØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ....
EØ ØØØØ ØØØØ Ø053 4352 4950 5349 544C 432Ø .....SCRIPSITLC
+ØØ FØ 9642 9642 2AØØ 16Ø3 1E24 FFFF FFFF FFFF B B*....$

FIGURE 5

system file, consult your Super Utility Manual for the proper procedures.

Finally, run Disk Repair and use the Repair GAT and Repair HIT options. The system files should now be accessible to the system, and the disk should boot (providing it was a booting disk to begin with) and behave normally--except that you've lost access to some of your files (Note: Don't use repair GAT on Multidos disks!).

If a nightmare occurs on a HIT or GAT sector, use Format without Erase to make the sector usable. Then run Repair HIT or Repair GAT as required. Again, don't use Repair GAT on Multidos disks.

RECONSTRUCTING A TRACK

If you just can't give up on salvaging your nightmare ridden sectors, no matter how rocky the road nor how unlikely your prospect of success, try this method.

DON'T Format without Erase--yet! Go into Super Utility's Memory Utilities section. Notice that item E is labeled Track to Memory. This function may be able to help salvage at least part of your lost data. The reason for this is that it performs a track read.

There are two different ways the FDC chip can be told to read or write information--sector or track. Normally the sector method is used, except during formatting. Track reads aren't as reliable as sector reads, because they dispense with some of the safeguards I discribed in the technical introduction.

A track read does have one advantage in dealing with farkled disks, however--namely, the FDC does not go into a tizzy if a sector ID is mangled or missing, or if a CRC does not compute. In fact, it doesn't treat ID's or CRC's as anything out of the ordinary. It reads in everything on the track as data--including gaps, formatting marks, file load format codes, and all.

As you may recall, the FDC won't complete a sector read if it encounters an ID error, or header CRC error. So by performing a track read you may get data into your computer that the FDC wouldn't look at during a sector read.

Assuming that your track read succeeds completely--and that's a pretty big assumption--you might think that you only have to do a track write to get the information back on the disk in usable form. Uh-uh! Here's where one of those little FDC kinks steps in to make a difficult situation even more difficult.

Believe it or not, if you do a track read of track A into a memory buffer, and track write that buffer onto track B, track A and track B usually don't end up containing the same data. That's because a track read reads it all in like it is, but there are certain bytes which a track write changes.

For instance, trying to write an F7 hex during a track write, will not put an F7 on the disk. Instead, it will write two CRC bytes, which might have any value. Trying to include any byte from F8 to FF hex in a track write, will result in the specified byte being written, but with an altered clock frequency. When the FDC later tries to read it during normal sector I/O, it will look like an address mark instead of a data byte.

I'll give you the general strategy for track read reconstruct, and then take a closer look at some of the details. First, of course, you do a track read of the nightmare track into a memory buffer. Then you separate the wheat from the chaff. That is to say, you look at all the stuff read into the

```
_   E3Ø0 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
HEXE31Ø FFFF CE0Ø 0000 0199 78FF FFFF FFFF FFFF
MEME32Ø FFFF FFFF FF0Ø 0000 0000 00FB E5E5 E5E5
   E33Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E34Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E35Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E36Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E37Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E38Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E39Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E3AØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E3BØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E3CØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E3DØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E3EØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
+ØØE3FØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5

   E4ØØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
HEXE41Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
MEME42Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 A4ØC FFFF
   E43Ø FFFF FFFF FFFF FFFF FFFF 0000 0000 0000
   E44Ø FE0Ø ØØØ5 Ø166 8DFF FFFF FFFF FFFF FFFF
   E45Ø FFFF FF0Ø 0000 0000 00FB E5E5 E5E5 E5E5
   E46Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E47Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E48Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E49Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E4AØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E4BØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E4CØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E4DØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E4EØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
+ØØE4FØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5

   E5ØØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
HEXE51Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
MEME52Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E53Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E54Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E55Ø E5E5 E5E5 E5E5 E5E5 E5E5 A4ØC FFFF FFFF
   E56Ø FFFF FFFF FFFF FFFF 0000 0000 0000 FE0Ø
   E57Ø ØØØ1 Ø1AA 49FF FFFF FFFF FFFF FFFF FFFF
   E58Ø FF0Ø 0000 0000 00FB E5E5 E5E5 E5E5 E5E5
   E59Ø E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E5AØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E5BØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E5CØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E5DØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
   E5EØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
+ØØE5FØ E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5
```

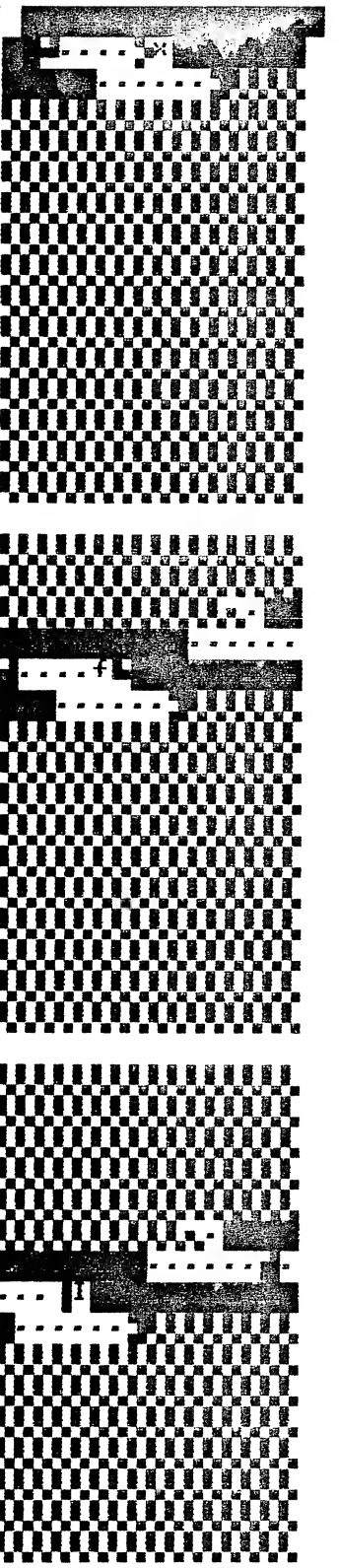


FIGURE 6

```
   E3ØØ FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
HEXE31Ø FFFF CEØØ ØØØØ Ø1F1 D3FF FFFF FFFF FFFF
MEME32Ø FFFF FFFF FFFF FFFF FFFF CBØØ FE11 F331
   E33Ø FC41 21E2 42CD 9A42 3EØ1 32E1 373A Ø242
   E34Ø 571E Ø4Ø1 ØØ4D CDAA 422Ø 7Ø3A ØØ4D E61Ø
   E35Ø 21E5 4228 69D9 2A16 4D55 7CØ7 Ø7Ø7 E6Ø7
   E36Ø 67Ø7 Ø784 5FØ1 FF4D D9CD 7542 3D2Ø 17CD
   E37Ø 7542 47CD 7542 6FØ5 CD75 4267 Ø528 EACD
   E38Ø 7542 7723 18F6 3D28 ØBCD 7542 47CD 7542
   E39Ø 1ØFB 18D5 CD75 42CD 7542 6FCD 7542 67E9
   E3AØ D9ØC 2Ø14 C53E Ø132 E137 CDAA 422Ø ØCC1
   E3BØ 1C7B D6ØA 2ØØ2 5F14 ØAD9 C921 F142 CD9A
   E3CØ 42CD 4ØØØ 76E5 7EFE Ø328 Ø8CD 33ØØ 23FE
   E3DØ ØD2Ø F3E1 C9C5 CDB2 42E1 C844 4DED 53EE
   E3EØ 3721 EC37 361B F5F1 F5F1 7EØF 38FC 3688
+ØØE3FØ D511 EF37 C5C1 18ØB ØF3Ø ØA7E CB4F 28F8

   E4ØØ 1AØ2 Ø318 F67E E65C D1C8 36DØ C91C 1FØ3
HEXE41Ø 17E8 4E4F 2Ø53 5953 5445 4DØD 17E8 4449
MEME42Ø 534B 2Ø45 5252 4F52 ØDEB 5F3Ø E5FF FFFF
   E43Ø FFFF FFFF FFFF FFFF FFFF FFFF FFFE CEØØ
   E44Ø ØØØ5 Ø1ØE 26FF FFFF FFFF FFFF FFFF FFFF
   E45Ø EØØØ ØØØØ ØØØØ 1FFB Ø5Ø6 5359 5333Ø 2Ø2Ø
   E46Ø 1FA9 ØD2A 2Ø2A 2Ø2A 2Ø4E 2Ø4F 2Ø54 2Ø49
   E47Ø 2Ø43 2Ø45 2Ø2A 2Ø2A 2Ø2A ØD2A 2Ø5Ø 524F
   E48Ø 5Ø52 4945 5441 5259 2Ø5Ø 524F 4752 414D
   E49Ø 2Ø2A ØD2A 2Ø43 4F5Ø 5952 4947 4854 2Ø28
   E4AØ 6329 2Ø31 3937 392Ø 2Ø2A ØD2A 2Ø42 592Ø
   E4BØ 5441 4E44 592Ø 434F 5250 4F52 4154 494F
   E4CØ 4E2A ØD2A 2Ø2Ø 464F 5254 2Ø57 4F52 5448
   E4DØ 2C2Ø 5445 5841 532Ø 2Ø2A ØD2A 2Ø41 4C4C
   E4EØ 2Ø52 4947 4854 532Ø 5245 5345 5256 4544
+ØØE4FØ 2Ø2A ØD2A 2Ø2A 2Ø2A 2Ø4E 2Ø4F 2Ø54 2Ø49

   E5ØØ 2Ø43 2Ø45 2Ø2A 2Ø2A 2Ø2A ØDØ1 144B 4ØØØ
HEXE51Ø ØØ37 4537 4537 4537 4537 4537 4537 4537
MEME52Ø 45Ø1 9ØØØ 45A3 45A3 45A3 45A3 45A3 45A3
   E53Ø 45A3 45A3 45A3 45A3 45A3 45A3 45E5 F53A
   E54Ø E037 214B 4Ø77 2CA6 28Ø8 2C1F 38ØF 2CB7
   E55Ø 2ØF8 3A4Ø 38E6 Ø42Ø AD95 FFFF FFFF FFFF
   E56Ø FFFF FFFF FCØØ ØØØØ ØØØØ Ø3FE ØØØØ Ø1Ø1
   E57Ø C2E2 FFFF FFFF FFFF FFFF FFFF FFFF FFFF
   E58Ø FFFF FFF9 CBØ1 ØØØØ 4E58 5A51 6D6Ø 6B62
   E59Ø 696C 676E 6568 636A 6174 7F76 7D7Ø 7B72
   E5AØ 797C 777E 7578 737A 7144 4F46 4D4Ø 4142
   E5BØ 434C 234E 4F27 494A 3F54 553F 575Ø 3252
   E5CØ 5339 5D5E 5E58 595A 5B24 2F26 2DDD 2B22
   E5DØ 292C 2D5A 4741 5A2A 4444 5Ø44 5644 585C
   E5EØ 543C 4E47 4C4C 5C57 3B73 6475 Ø7ØØ ØB02
+ØØE5FØ Ø9ØC 696B 7C61 6E64 6E7Ø 1574 6E1Ø 7Ø7C
```

FIGURE 7

buffer and try to figure out what represents genuine user data, and what represents genuine garbage. Then you get all the good stuff down on paper. Finally, you reformat the nightmare track and use Zap's Modify Memory mode to reenter and save the data, sector byu sector.

Now for the gruesome details: When you do the track read, make sure the drive is configured correctly for the disk. It is especially important to do the track read in the proper density. When you ask Super Utility to do the actual read, it will request the drive and track number. After you provide that information, it will ask for the start of the buffer into which the track will be read. It's usually a good idea to default by pressing <ENTER>. Super Utility will select a safe buffer.

Finally, you will get the prompt, 'Sync to ID marks ?'. Super Utility asks this because there are two ways the FDC may be instructed to track read--with ID sync or without. If you elect to sync, the FDC will try to sync with ID fields whenever it encounters them. If you elect not to sync, the FDC will just read everything it can, from the first byte on the track, making no attempt to resynchronize when it encounters an ID field.

One would expect that track read with sync would invariably produce more reliable results than track read without. Actually, the reverse seems to be the case. You should try several track reads of each kind on your nightmare track. Each time, inspect the result carefully. You'll find that often even two track reads of the same kind do not produce identical results. Choose the with or without sync option that seems to work best for your problem.

When you have finally decided that the material you've track read into the buffer is as good as it's going to get, it's time to start in on the wheat and chaff thing. Figure 6 shows the result of track reading a formatted Model I TRSDOS track--a "virgin" track which contains no data other than formatting information and "fill" bytes. Figure 7 shows a similar track after files have been saved on it.

Looking at the first figure, you will notice the track is buffered in memory starting at location D400 hex. It is laid out in the manner described earlier. First there a rather long gap. The track read may pick up some garbage at the beginning of the track as well, but in this case, the read was clean. Skip through the bytes until you come to the first FE hex in the buffer (in the example, it's displayed at memory location D415 hex). That marks the beginning of the first physical sector on the. disk.

Following the FE is the byte sequence 06 00 00 01. The first three bytes indicate that the track number is six, the head number is zero, and the sector number is zero. The one at the end of the sequence is the length code. In IBM standard (which TRS-80 DOS's use), one means a 256 byte sector. The next two bytes (D6,4A) are the CRC for the header-subfield.

Next we have another gap consisting, in this case, of 12 FF's and six zeros, for a grand total of 18 bytes. Then comes the Data Address Mark, FB.

After the DAM comes the data. The track has not had any files saved to it yet, so the data field contains what ever "fill bytes" DOS used for formatting. As you see, in this case the fill byte is E5, and there are exactly 256 of them.

E5 hex is the "worst case" data pattern for single density disk I/O. That means that if a physical disk sector is in marginal condition, a series of E5's is the data pattern most likely to cause an error. The worst case pattern for double density is the byte pair 6D B6.

Most DOS's fill data areas with worst case patterns when they format. This is so you'll become aware of error prone disks before you entrust vital data to them.

After the E5's comes the byte pair, A4 0C. This is the data CRC. Finally, we have another 18 byte gap, another FE sector ID, and all the shenanigans start again with sector five. And so it goes.

If you examine the second figure, you'll notice it's pretty much the same as the first, except for the sector data, and the data CRC's.

That should give you an idea of what a track-read track should look like when nothing has been munched. Now suppose you track read a nightmare track. When you examine the buffer, you should see a pattern like the one in figure 7--up until the point where the disk got clobbered. If you're lucky, there'll only be a few bytes of garbage, and then normality will resume. Unfortunately, here's what is much more likely to happen: after the FDC passes over the garbage, it will be out of sync with the remaining data on the track. That is to say, when it tries to read the garbage, it will lose some bits, or pick up extra ones. Thereafter, even if the FDC reads all the subsequent bits faithfully, they will be not grouped into bytes correctly. In effect, the data will become shifted.

Since shifted data looks about as much like garbage as genuine garbage, this can make it difficult to figure out what's going on in the track. Here's where Super Utility's decryption features come to the rescue. That's right, all that encrypting-decrypting stuff isn't just for people who want to play with secret messages. It can help you save your disks!

What you have to do is retry the track read operation several times, with and without sync. When you're satisfied that you've got the best results that you're going to get, examine various parts of the garbagy looking stuff, and use the decryption facility to shift through all eight possible phases.

The way to do this is well covered in your manual. Briefly, what you do is go into Display Memory and look at a trashy part of the track read buffer. Make sure you're in the paging mode (not the modify mode), and press <@>. This makes the DCR (for DECRYPTION) prompt appear near the bottom left hand corner of the screen. Then enter <:>. This will cause all decryption changes to be visible on both the hex and ascii portions of the display.

press <@> again to get the DCR prompt back. Then enter SL1 (for Shift Left 1 bit) or SR1. If the display that results looks like what you're hoping for, congratulations--you've probably struck pay dirt! Do a screen print, or, if you don't have a printer, copy the data by hand. If a shift of one doesn't do the trick, try again with shifts of two, three, ..., through seven (Note: for a fuller treatment of the decryption mode, see the Undocumented Features Chapter).

Even if you do manage to capture meaningful data, you'll have to know what it looks like in order to recognize it. Of course, there's always the chance that you can spot true data by the pattern of gap, header ID, gap, DAM, 258 bytes of what ever (the 256 byte data field plus two bytes of CRC), gap, etc. That's why I detailed the pattern. But things are a lot easier when you know what the data looks like.

If the track contained ASCII word processing material or the like, half the battle's already won. If it contained a machine language program, it would help if you knew some of the code. If such is the case, you may find what seems to be a sequence of the program that you're looking for, except that extraneous bytes are thrown in. These could be load file format codes. For more information on load file format codes, see the section of this book which deals with patch sectors. Also, your favorite DOS manual may shed some light on the subject. My own introduction to the topic was via the technical section of my LDOS manual.

If nothing you do makes your lost data appear in recognizable form, you're just out of luck. However, if you do manage to tune the data in, your work's just begun. Print it all out in sequence. Isolate the sector (or sectors) which were farkled by the nightmare. It will be missing the header subfield

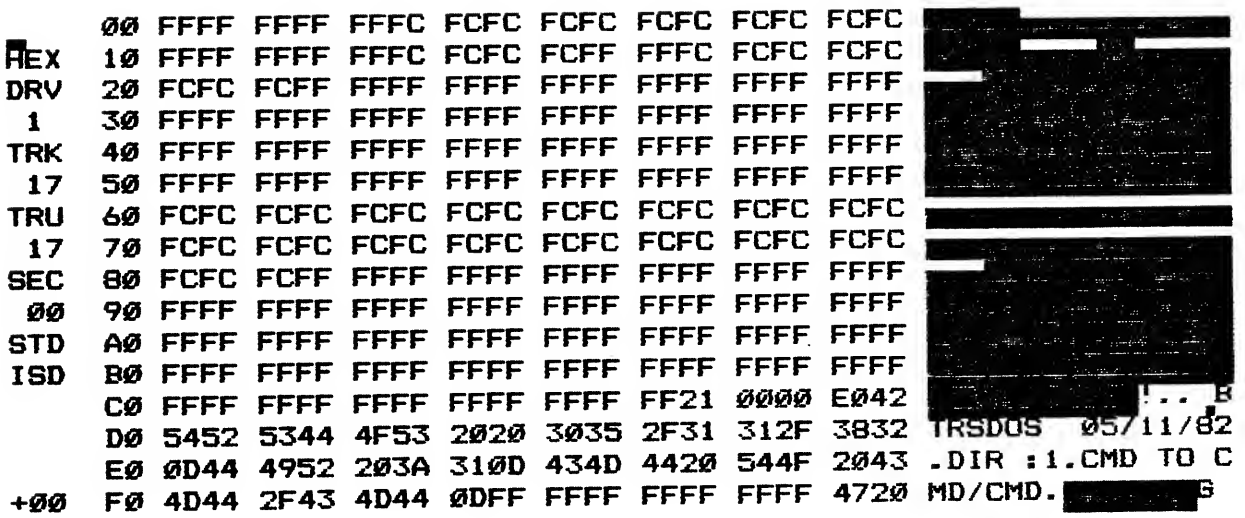

ØØ FFFF FFFF FFFC FCFC FCFC FCFC FCFC FCFC
HEX 1Ø FFFF FFFF FFFC FCFC FCFF FFFC FCFC FCFC
DRV 2Ø FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF
1 3Ø FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
TRK 4Ø FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
17 5Ø FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
TRU 6Ø FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC
17 7Ø FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC
SEC 8Ø FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF
ØØ 9Ø FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
STD AØ FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
ISD BØ FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
CØ FFFF FFFF FFFF FFFF FFFF FF21 ØØØØ EØ42
DØ 5452 5344 4F53 2Ø2Ø 3Ø35 2F31 312F 3832 TRSDOS ØS/11/82
EØ ØD44 4952 2Ø3A 31ØD 434D 4420 544F 2043 .DIR :1.CMD TO C
+ØØ FØ 4D44 2F43 4D44 ØDFF FFFF FFFF FFFF 4720 MD/CMD.

FIGURE 8

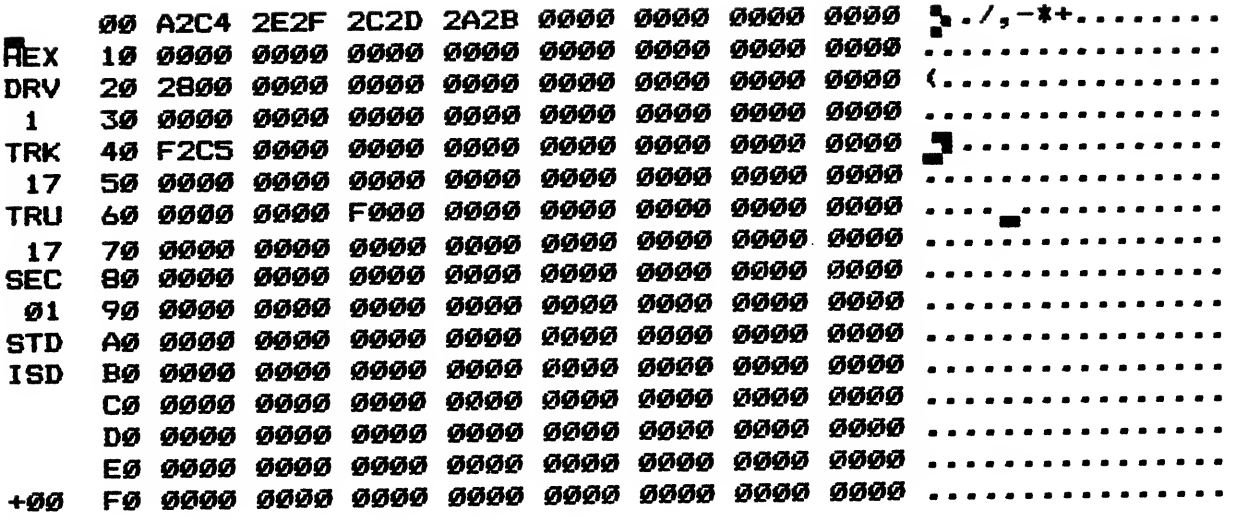

ØØ A2C4 2E2F 2C2D 2A2B ØØØØ ØØØØ ØØØØ ØØØØ
1Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
2Ø 28ØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
3Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
4Ø F2C5 ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
5Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
6Ø ØØØØ ØØØØ FØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
7Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
8Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
9Ø ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
AØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
BØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
CØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
DØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
EØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
FØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
REX
DRV
1
TRK
17
TRU
17
SEC
Ø1
STD
ISD
+ØØ

FIGURE 9

(or part of it,) or the DAM, or else the header-subfield CRC will be wrong.

If an ID or DAM is missing, study the printout and figure out where it belongs. Part of the data will probably be gone too. Mark the printout to indicate what's missing from where. If you reconstruct the sector correctly on paper, you'll get it right on the disk too.

Now you're ready to start setting up the disk. Since it's always safer to work with backups, backing up the farkled disk should be the first thing you do. Use Super Utility to make the backup, and just S>kip the problem sector(s) when Super Utility prompts you with the mini-menu.

Then go into Zap and display the nightmare sector on the backup. Use the modify mode to type in the sector data. If more than one section was nightmared, repeat as necessary--and good luck!!!

DIRECTORY REPAIRS

After all else has been done, it's time to check, and if necessary repair, the directory. Simply go to Super Utility's Disk Repair module (item five on the main menu) and select option nine, "Check Directory." If Super Utility reports zero errors, that's it. If Super Utility reports a HIT or GAT error, run Repair HIT Sector or Repair GAT Sector.

In Repair GAT Sector, you will be asked whether you want to fix just the allocation table, or the entire sector. Usually, just fixing the allocation table will be sufficient. Fixing the entire sector also replaces the disk name, date, master password, and auto commands with Super Utility's defaults. You'd want to do that is if those fields contained garbage. You'd know that was the case if Super Utility displayed trash whenever it claimed to be reporting a disk's name and date.

WARNING: Do not attempt to repair the GAT on a MULTIDOS disk. This is a new DOS and is not yet fully supported. Attempting a Repair GAT operation on a MULTIDOS diskette could severely aggravate the problem you're trying to solve!!!

If Check Directory, or any other operation, results in a message to the effect that the directory can't be located, you probably have to run Read Protect Directory. Since, when you have this problem, Super Utility can't locate the directory on it's own, it can't read protect it without your help. It will ask you for the track and sector the directory starts on, and its length in sectors.

35-40 track diskettes normally have the directory on track 17 decimal. 80 track disks usually have the directory on track 40, to keep it in the center of the action. NEWDOS-80 and Doubledos may have it elsewhere. Directories are usually one

track long--that's ten sectors for single density, and 18 for double. Again, don't take anything for granted with NEWDOS-80.

If you're not sure about the directory's location and length, go Zapping through the disk until you find it. You've already seen what a typical directory sector looks like. That was a file-location sector. The first sector in a directory is the GAT. It contains the disk name, date, master password, and auto command in its last several bytes. That should help you recognize it. Figure 8 is a sample GAT. The second directory sector is the HIT. HIT's look pretty garbagy. Figure 9 is a typical HIT sector.

After you have located and read protected the directory, rerun the Check Directory program, and run any repair programs indicated.

There is one case when Check Directory will be unable to find the directory even though the disk is a perfectly good one, and that is when you try to read a Model I TRSDOS disk (single-density) on a Model III. The problem is that the DAM used by Model I TRSDOS for the directory track is not one that the Model III can read! By using "Read Protect Directory", you can make the Model III read your Model I disk directory, but the process opens up a new can of worms: now Model I TRSDOS will not be able to find the directory!

In order for Model I TRSDOS to be able to use this disk again, you must boot Super Utility on a Model I, put in the disk which you read-protected on your Model III, and repeat the process. This time the directory will be re-written to the disk with the correct DAMs.

Chapter V
UNDOCUMENTED FEATURES

Super Utility has a few features that, for some odd reason, Kim hasn't chosen to reveal to the world at large--till now! These features include several "control-key" functions, joystick input, and a new encryption/decryption feature.

There is a kind of control key arrangement that is used to activate some of these features. The "control key" consists of the letters "K," "I," and "M," which you must press simultaneously. Here is a list of KIM features:

1) KIM + SHIFT + up-arrow and down-arrow (all simultaneously) = wizard and serial number.
2) KIM + "T" = trace enable.
3) KIM + "D" = trace disable.
4) KIM + "J" = joystick enable

1) Holding down the control combo <K>, <I>, <M>, and pressing the SHIFT, <up-arrow> and <down-arrow> keys results in a display of two items: a) the name of the wizard who created Super Utility, and b) the serial number of your copy.

2) The control combo and <T> turns on a trace function similar to TRSDOS's.

3) The control combo and <D> turns off the trace function.

4) The control combo plus <J> activates joystick control. Yes--you can control Super Utility with a joystick!

First of all, you need one of the new Alpha joysticks with the 5/4 switch on the connector. To use it with Super Utility, put the switch in the five position.

Second of all, you should know that this is not intended as a practical feature--just as a fun surprise to show that even a serious program like Super Utility can have a lighter side.

Joystick control will not work with certain nonstandard TRS-80 configurations. For instance, old LNW interfaces return a different value from "unused" ports than do standard Tandy expansion interfaces. This makes joystick control impossible with such configurations. If pressing the control combo plus <J> results in Super Utility's flaking out, you can't use the joystick feature with your system. Reset the computer and forget about control <J> and joysticks.

The majority of TRS-80 systems should be Alpha joystick compatible. To enable Super Utility joystick control, turn off you computer and connect the joystick according to the instructions which came with it. Make sure the switch is in the "5" position. Now power up your computer and boot in Super

Utility. You should be in the Master Menu. Press control <J>.

Move the joystick diagonally to the upper right and hold it there. You will notice a changing character near the upper right hand corner of the display, next to the alive character (if you don't know what an alive character is, you will find an explanation in the MISCELLANEOUS chapter.

The changing will increment through all displayable values. Somewhere along the line, it will get to the realm of numbers and letters. Notice that you can reverse the action and start the character decrementing by moving the joystick to the lower left position.

This special controllable character indicates the value to be selected by the joystick and will be used for all joystick data entry. When you finally reach the value you want, you can launch it down to Super Utility's normal prompt-command line by moving the stick to the lower right. Because of its proximity to the alive character, we shall call this joystick character the Joy Of Life Launch Indicator, or JOLLI.

Suppose we want to perform a Verify Sectors operation. Starting from the main menu, press the joystick's red "fire" button. This will put you in the Zap menu (pressing the "fire" button always has the same effect as pressing <ENTER>). Move the joystick to the upper right position. Release it when when the JOLLI reaches "2." If you pass two, move the stick to the lower left to go back.

When you succeed at getting the JOLLI to indicate a two, release the stick to let it return to the center position. Then move it to the lower right to launch the JOLLI character to Super Utility's prompt-command line.

Some joysticks seem a little difficult to control properly; so it may take some experimentation and practice before you get the hang of setting the JOLLI to the value you want and launching it. However, with a little practice it becomes easy. In the meantime, you can always "backspace" over any errors which you launch to the command line by moving the stick to the left. In fact, moving the joystick to any of the four "cardinal" positions will have the same effect as pressing the corresponding arrow key.

When you have "2" on the command line, enter it by pressing the "fire" button . Super Utility should now be presenting you with its 'DRIVE, TRACK, SECTOR' prompt. Suppose you want to start the verify at drive zero, track three, sector one. Set the JOLLI to zero and launch it.

We have now specified the drive number. The next thing we have to do is enter a comma (","). Do so by positioning the stick to the upper left. Now we must select track three. Set the JOLLI to three in the (by now) accustomed manner and launch it. Enter another comma by moving the stick to your upper left, and then select sector one with the JOLLI. Remember to backspace as necessary to correct errors. When the command line is correctly

set up, enter it by pressing "fire." Pressing "fire" again will default on the number-of-sectors question and bring about the desired verify operation.

Here's a tabular summary of the joystick controls:

Up	=	up-arrow
Upper-right	=	increment JOLLI
Right	=	right-arrow
Lower-right	=	launch
Down	=	down-arrow
Lower-left	=	decrement JOLLI
Left	=	left-arrow or backspace
Upper-left	=	comma ","
Fire button	=	<ENTER>

Note that there is no way to simulate the <SHIFT>, <BREAK>, or <CLEAR> keys with the joystick. These keys will always have to be entered through the keyboard. If you are really determined to enter even these instructions via the joystick, you may turn the joystick upside down. Then, holding it by the base, use the tip of the stick to press the <BREAK> or <CLEAR> key.

Pressing <SHIFT><BREAK> can be a problem. If you invert the joystick and closely inspect the base, you will notice little rubber feet on the bottom. The two that will be on the left when the joystick is in its normal right-side-up position are just about the same distance apart as the TRS-80's <SHIFT> and <BREAK> keys. It may be possible, by very careful manipulation, to use those two feet to press the <SHIFT> and <BREAK> keys simultaneously, without pressing any of the intervening keys. However, I have yet to succeed at this maneuver. If you can afford another $40.00, it would probably be better to buy a second joystick to use on the <SHIFT> key.

The last undocumented feature I have to share with you is the new encryption/decryption function that I mentioned. (Heretofore, I shall simply use the word "cryptography" instead of "encryption/decryption.") Before describing the new feature, I will present a brief overview of the Super Utility cryptography features.

The cryptography mode may be entered from either Zap's sector-display mode, File Utilities' Display File Sector mode, or Memory Utilities' memory-display mode. Both the memory and sector display modes present a screenful of information--the former about a disk sector and the latter about a 256 byte memory block.

The screenful contains hex information on the left, and its ASCII equivalent on the right. Remember that these display modes each have two sub-modes--the paging mode and the modify mode. In the paging mode, there is a single cursor at the top left hand corner of the screen, in the "margin." You get into the modify mode by pressing <M> when in the paging mode. You'll know you're

in the modify mode when you see the two cursors, one in hex area and one in the ASCII area, of the main display. To enter cryptography, you must be in the "paging", not the Modify mode.

To enter cryptography, press <@>. A 'DCR' (for DeCRypting) prompt will appear on the screen. Super Utility is now ready to manipulate the display for you. It does so by performing any of a number of logical or arithmetic operations on the display material.

For example, you may ask Super Utility to add "1" to every byte in the display. The display will be updated accordingly. There is an important distinction to grasp here. Only the display is being altered. The actual memory or disk sector being displayed is not. You may perform a number of cryptological manipulations and then cancel the cryptological display. You will see that the target Memory-block or sector still contains the original data. There is one exception to this. That is what the new feature is all about. We shall get to it shortly.

Let's go through a sample cryptography session. Get into Memory Utilities' memory display routine. Use F000 hex as the target area. F000 is well above Super Utility's code and buffers, and any changes we make there should have no effect on the program's operation.

The first thing to do is to fill the memory block with contents which will make the effects of any shifts or rotations readily apparent. If the default data entry mode (as revealed in the upper left of the display) isn't already hex, make it hex by pressing <H>. Then press <M> to enter the modify mode. The cursor should be over the first byte of the sector.

Type "55." The first byte in the displayed block should now be 55 hex, and the cursor will have advanced to the second byte. Use the left arrow to move the cursor back over the first byte. Press <P> for Propagate. Then type "FF." "FF" is hex for 256. All 256 bytes in the memory block should now be 55. Press <ENTER> to leave the modify mode and return to the paging mode. Note that so far we have used only standard memory-modify procedures, not cryptography features.

Press <@> to get the 'DCR' prompt. The first thing to do is to tell Super Utility whether you want the results of your cryptography experiments to appear in the ASCII portion of the display only, or in the hex side as well. If you were looking for a hidden message, such as a copyright notice, you would only be interested in the ASCII. If you were looking for disguised machine code, you'd want to see the results of your alterations in hex as well. For the purpose of this demonstration, we'll select the latter mode. To do so, enter colon <:>.

Notice that each time you enter a cryptography command, the 'DCR' prompt vanishes and Super Utility is ready to receive normal paging commands--but the display is still in whatever cryptography mode was last selected. For instance, press <@> to get 'DCR' back. Now tell Super Utility to shift the display two

bits to the right by entering "SR2" (for Shift Right 2).

As you see, the display changed in accord with your command, though the 'DCR' prompt vanished and Super Utility is ready to accept normal paging commands. From now on, all memory and sector-displays will be shifted two to the right, until you cancel this condition. As a reminder, the letters SR2 will remain near the bottom left hand portion of the screen.

You may override the shifted display by entering different cryptography command. Or you may cancel all cryptography and make the screen revert to normal by keying <@> and pressing <ENTER> without pressing any other keys.

You may use parallel syntax to shift or rotate right or left anywhere from zero to seven bits. To AND every display byte with a value, get the 'DCR' prompt and then enter <A> (for "AND") followed by the value. For example, you could enter "A1" to AND the display with one. Use the same syntax for OR and Exclusive OR using "O" or "X" instead of "A." To add or subtract a one byte value from every byte on the display, use a "+" or "-" followed by the value to be added or subtracted. For instance, "+5" would add five to every byte in the display.

There is also an automatic, or movie, mode. This only works for all DCR modes. Answer 'DCR' with an up-arrow for automatic increment or a down-arrow for automatic decrement. You may follow the arrow with a number between 0 and 255. This determines a timing factor which controls the display change rate. Try it, it's fun!

You can use cryptography for encrypting as well as decrypting. For instance, suppose you want to create a sector on your disk containing your name, address, a copyright statement, and perhaps some other information--all hidden. First find an unused sector--you can use File Utilities' Disk Allocations options to do so. Then use Zap's Display Sectors to display it.

Go into the modify mode and zero the display. An easy way to do that is to use the <P> command to propagate a series of zeros through the sector. Then type <SHIFT><ENTER> followed by <A> to change the input mode to ASCII. Type your message into the sector display. When you are done, press <ENTER>. Answer the next prompt by entering <U>. This will save what you've accomplished so far to disk (in non-encrypted form) and return you to the paging mode.

Now you may start encrypting. Suppose you decide to hide the message by adding 5 to every byte. Press <@>. Then answer the 'DCR' prompt by entering "+5." If only the ASCII side of the screen changed, select the full screen update by getting the 'DCR' prompt and entering <:>.

All that remains now is to save the sector to disk in this disguised format. But remember, the cryptography features discussed so haven't actually changed the sector-only the display. In other words, even if you were to resave the sector, it would still contain the same data.

In order to make the change "real," use the <@> key to get the DCR prompt. Then enter <!> (Exclamation mark). The "!" is the previously undocumented feature. It will cause the actual sector buffer to be altered so as to conform to the display. Saving the sector now will result in your message being stored on the disk in the encrypted form. If, instead of the sector-display mode, you had been in the memory-display mode, the encrypted material would have been transfered to memory as soon as you entered the <!>.

Chapter VI
MISCELLANEOUS

When you boot in Super Utility, the first thing that happens is that the "title page" is displayed on the screen. Then, before the bulk of the program is loaded, a quick memory test is performed on your system. If a memory fault is detected, a message to that effect is displayed and the program load is aborted.

You may think it's a little snobbish of Super Utility to refuse to inhabit a computer that flunked its memory test--but the Surgeon General has determined that running Super Utility in a flaky system is hazardous to your disks. The memory test lasts only a few seconds, but on some Model I's, this may be long enough for the disk drive to time out. So don't be alarmed if your drive does a little extra clicking when you load Super Utility. It's all for a good cause.

When the memory test has been passed, the rest of Super Utility is loaded. During this process, a sort of arrow moves across the screen from left to right, top to bottom. On the Model I, it looks a little like a rocket ship leaving behind a vapor trail of dashes. On the Model III, it's a pointing finger. Each dash represents a sector that has been loaded in. At the end of the arrow's trip, it hits a graphics block. When the arrow hits the block, the load has been completed.

By watching the arrow's progress toward the block, you can tell how much of Super Utility has been loaded at any given moment, and how much remains to be loaded. If the arrow tends to slow or get stuck at any point, you know that your system has trouble reading that part of the disk. In that case, it may be time for some preventive maintenence--such as head cleaning, speed adjustment, and possibly even head alignment. Maybe, just maybe, you need a replacement Super Utility disk.

After Super Utility has been loaded, it performs one last disk access before quieting down. During that access, it loads the hard configuration and patch sectors.

Many computers utilize something called interrupts. TRS-80 disk systems are among those that do. This means that 40 times every second (30 on the Mod III - Ed.), the TRS-80 abandons what it's doing and executes special subroutines. These are called service routines. They help the system keep on top of situations which require frequent attention. For instance, the memory locations which keep track of time in a regular disk operating system (not Super Utility) are updated during service routines.

There are certain timing-sensitive operations, such as tape I/O, which would be thrown off by interrupt-interruptions. Certain stages of disk I/O also require freedom from interrupts. Interestingly enough, other stages of disk I/O can't take place without them. To accommodate such requirements, the Z-80 microprocessor has been given the ability to ignore (disable) interrupts or respond to (enable) interrupts, according to directives in its programming. During normal operations, interrupts are enabled a very high proportion of the time--what amounts to nearly constantly.

You have probably noticed that at nearly all times, there is a moving graphics block near the upper-right hand corner of Super Utility's display. This is called an alive character. The movement of this block is attended to during interrupt processing. Therefore, when the block is moving, it shows that the service routine is working, or alive. That's why it's called an alive character.

The alive character probably won't be of much practical use to you. Some DOS's (LDOS and VTOS) offer it as an option. If a program bombs and the system hangs, knowledge of whether the service routines are still running may help you diagnose the cause of the flake-out. An alive character gives you that information. However, it's very unlikely that you'll find yourself debugging Super Utility.

There is one way in which the alive character may be useful to you, though. It has to do with the fact that there are two ways of getting Super Utility to abort an operation. Usually <BREAK> will terminate any activity. However, there are a few instances in which only <CLEAR> will do the job.

Generally, <BREAK> works when interrupts are occuring (remember, that's nearly always), and <CLEAR> may be used when interrupts are disabled. So the motion (or lack of motion) of the alive character will let you know which of those two keys is appropriate.

An exception to this principle is that <CLEAR> is always used to abort a screen print and empty the print buffer. Also, you can try using <CLEAR> to terminate an activity when you don't want to be returned to the menu.

Some of you may have wondered what software (other than Super Utility itself) Kim uses during his hours at the keyboard--what are the software tools of his trade? I, at least have been curious about this, so I asked him. Kim uses the LDOS disk operating system, the EDAS editor/assembler, and Jake Commander's Macromon (Shadow) monitor.

Another thing you may have wondered about is the unusual disk addressing scheme used by double density NEWDOS-80 and Doubledos. Why are the "relative tracks" different from the physical ones? This system is called Disk Relative Sectors or

DRS.

There is a bit of a paradox involved here. This apparent complication really involves an attempt to make things simpler and more transparent. It stems from the fact that single density is divided into two grans of five sectors each, but double density tracks contain three six-sector grans. DRS lets you pretend that everything's always the same. In other words, you can act as if double density disks still consisted of ten sectors per track.

As an example, let's consider a double density NEWDOS-80 disk with a double density track zero. In reality, every track contains 18 sectors numbered zero through 17. I will call such a real track, which consists of 18 sectors, a physical track or p-track.

NEWDOS-80 will take the first ten sectors (0 through 9) of the first p-track zero and call them track zero. I'll call them relative track (or r-track) zero. That leaves eight sectors (10 through 17) in p-track zero out in the cold. So NEWDOS takes those eight sectors and groups them with the first two sectors on the next real track, and calls this 10 sector aggregation track two (that's r-track two to you).

Notice that the first r-track in our example occupies part of one p-track. The second r-track spans a real track boundary and occupies parts of two separate real tracks! Here's a map of how the first several p-tracks would be divided up into r-tracks. All numbers are in decimal:

Physical Track	Sector	Relative Track
0	0- 9	0
0	10-17	1
1	0- 1	
1	2-11	2
1	12-17	3
2	0- 3	
2	4-13	4
2	14-17	5
3	0- 5	
3	6-15	6

3	16-17	7
4	0- 7	
4	8-17	8
5	0- 9	9
5	10-17	10
6	0- 1	
6	2-11	11
6	12-17	12
7	0- 3	
7	4-13	13
7	14-17	14
8	0- 5	
8	6-15	15
8	16-17	16
9	0- 7	
9	8-17	17

Now suppose that to keep up appearances of normality, the directory is placed on r-track 17. It's actually on p-track nine. But NEWDOS-80 doesn't want us to be concerned with that. The idea is that when we want NEWDOS-80 to read a certain part of the disk, we tell it where to look in r-tracks and r-sectors. NEWDOS-80 will do all the conversions interally, and look at the correct real track.

If you have a double density disk with a single density track zero, NEWDOS-80 starts relative track zero at real track one, sector zero. So a 35 track diskette will have 36 actual tracks, since NEWDOS-80 doesn't count the single density track.

If all that's not complicated enough, NEWDOS-80 also lets us vary the number of sectors per gran (or lump, as NEWDOS-80 calls them) instead of sticking to the conventional five.

And if you have a double density system, it also allows the following combinations of formats:

Method	Track zero	All other tracks
1	single density	single density
2	single density	double density
3	double density	single density
4	double density	double density

As you can see, NEWDOS-80 allows all possible combinations. But Super Utility DOES NOT support them all. If you want all of Super Utility's features to work on your NEWDOS-80 disks, you must use standard configurations. Method (3) is not supported. Also, you will have to use grans which consist of five sectors.

For operations which are not file oriented, or don't in any way involve the directory, this restriction isn't too important. For such operations, configure Super Utility as if you were using another DOS. If the disk has format method (1), use "A" (Model I TRSDOS). If you used method (2), use "F" (double density Model I DOSPLUS). If you used method (4), use "G" (double density Model III DOSPLUS).

By the way, if you use format (2), NEWDOS-80 does another trick. It doesn't use or count physical track zero at all. When you boot the disk, p-track zero, sector zero will be read loaded into memory. But this is done by the ROM's bootstrap routine, not NEWDOS-80. As far as NEWDOS-80 is concerned, p-track zero doesn't exist. It starts r-track zero, sector zero, at p-track one. It even maintains a duplicate boot sector at p-track one, sector zero.

Customers often phone Breeze/QSD with questions about Super Utility. Most of the time, their questions can be answered by referring the customer to a page in the Super Utility manual. I blush to admit that I've made a couple of those calls myself.

Judging by the calls Kim receives, one of the most ignored parts of the manual is Note 6. It explains that when working with NEWDOS-80 or Model III TRSDOS diskettes, you must specify an exact track-count. The track-count required is obtained by subtracting one from the real track-count. Note 6 is quite explicit. Please read it if you use Super Utility on Model III TRSDOS or NEWDOS-80 disks.

While I'm on the subject, here's a follow-up warning. Remember, if you've configured 'SAVE=N', other aspects of the configuration table may not remain the way you left them. Therefore, you may have configured Super Utility with a correct track-count, but Super Utility may no longer be using it. Therefore, if you tend to keep 'SAVE=N', it's a good habit to use DOS specifiers and track-count overrides when working with

Figure 11

```
E>ncode or D>ecode ? d
Filename ? su6a/l
Reading    Drive 1, Track  40, Sector  17,
SU6A/L
Access     =FB2DH.
OM         =FB2DH.
Update     =E32FH.
MU         =E32FH.
                Key <ENTER>
```

TRSDOS III and NEWDOS-80 disks.

As an example, suppose you want to check the directory of a 40 track Model III TRSDOS disk. The disk is in Drive 0. When Check Directory asks you to enter the Drive(s), instead of entering "0," enter "0B=40." B is the DOS specifier for TRSDOS III and 39 is the real track-count. Now, regardless of what is in the configuration table, Super Utility will be able to handle things properly.

Here's a another warning about Model III TRSDOS. System files are not logged into the directory. This has some unpleasant implications. For instance, suppose Zap's Verify Sectors routine reports a bad sector on a TRSDOS III disk. You then use File Utilities' Sector Allocation program to see if that sector is allocated. It's possible that Super Utility will report the sector as unallocated, even though in reality the sector contains a system file. You might then Format Without Erase the disk, selecting the S>kip option whenever Super Utility balked at the bad sector. If you did so, you would probably end up with a disk which wasn't functional.

One preventative measure would be to follow the directions in your Super Utility user's manual, note 16. This will tell you how to create normal directory entries for the system files. Another solution would be to use the technique described in note 14 to look at all the system files on one of your TRSDOS III disks. Make a note of all the sectors they occupy. The system files should be located in the same relative positions on any other TRSDOS III disks you have, especially if they're all descended from the same master disk. Keep the list handy. When Super Utility tells you a sector is unallocated, cross check it with your list of system sectors to be sure.

Another peculiarity of Model III TRSDOS is the way it treats passwords which encode to zero. It changes them to something else. A result of this is that when you use File Utilities' Compute Passwords, the password it gives you might not work. This will happen when the password happens to be one which encodes as zero.

If you should get a password that doesn't work, just use Purge's Remove All Passwords utility. Or use Zap to change the individual file's password in the directory. Start by looking at the disk's directory sectors until you find the entry for the target file. Remember, each directory entry occupies two lines of Zap's display. The encoded hash value should be the first two bytes of the second line of the file's entry. If Super Utility gave you a password which didn't work, you should find that those two bytes are zeros. Use the modify mode to change either or both of them to anything else. Then go back and rerun Compute Passwords. The password given this time will work.

Figure 12

```
E>ncode or D>ecode ? d
Filename ? su6b/l
Reading    Drive 1, Track  40, Sector  17.
SUB6/L
Access     =9642H.
           =9642H.
Update     =9642H.
           =9642H.
                Key <ENTER>.
```

While we're on the subject of Super Utility and passwords, here's something else you should know. Figure 11 reproduces the input and output of a typical session with File Utilities' Compute Password.

As you can see, both the update and access passwords are decoded for us. The format of the output is as follows: first the filename is displayed in uppercase. Below that is the word "Access" followed by some spaces and an equal sign. To the right of the "=" is the encoded access password in Hex. On the next line is the decoded access password, followed by some spaces, and equal sign, and a repetition of the encoded access password.

The format is repeated for the update password. The decoded update password is displayed once, while the encoded update password is displayed twice.

This is all fairly clear. But now suppose we run Compute Password on a file which doesn't have any passwords. The display would look something like the one shown in figure 12. The passwords have been displayed as a series of blank spaces--which is what they actually are. 9642H is the way a blank password encodes in the directory. But if you haven't used Compute Password before, the display can be a little confusing. You might think that Super Utility is trying to tell you that 9642H is the decoded password. So, if Compute Password ever gives you an answer that looks like the one in figure 12, remember--it means the passwords are blank.

As this book is being prepared, Radio Shack is starting to publicize its new double density board for the Model I. At this time, no one seems to know whether or not it will be compatible with the standard set by Percom and more or less followed by LNW and Aerocomp (Editor's note: now we know that it isn't). Therefore, another unknown fact is whether the current version of Super Utility will be compatible with the Radio Shack board (ditto, but an update will be made available later this year.-Ed.).

Judging by the calls and letters Breeze/QSD gets, there is a certain mistake which people make rather frequently--one that can completely crash the program. What they do is overwrite part of Super Utility with other data.

A common time for this to happen will be during the use of Memory Utilities--especially Sectors to Memory and Track to Memory. When asked to specify a buffer, they will choose one too low--one right in the middle of Super Utility itself. When the track or sectors are read in, they overwrite a vital part of Super Utility. The crash may materialize immediately, or later on when a different program module is used.

The best way to avoid this is to just press <ENTER> and default when Super Utility asks you to specify a buffer location. Super Utility will then automatically choose a safe area of memory and inform you of its location. Remember, when in

doubt, default!

Extending a disk means adding tracks to it without erasing the data already on the disk. For instance, you can extend a 35 track diskette to 40 tracks. Elsewhere in this book, I discribed a very awkward method of extending disks using Format without Erase. Here's the easy way to do it.

Go into Super Utility's Disk Format section and select Standard Format. Answer the "Name," "Date," and "Password" prompts as you please. Answer the "Use Configuration ?" prompt by entering <N>. This will result in your being prompted with "DOS, Tks, Dir, St Tk ?"

You are being prompted to enter the DOS specifier, the track-count, the directory track number, and the starting track. Suppose you have a 35 track Model I TRSDOS disk on drive one which you want to extend to 40 tracks. Its directory is on track 17. Follow the instructions in the preceding paragraph. When Format gives you the "DOS, Tks, Dir, St Tk ?" prompt you will enter "A, 40, 17, 35."

"A" is the DOS specifier for Model I TRSDOS. "40" is the number of tracks you want the disk to end up with. "17" is the number of the directory track. The disk, to begin with, has 35 tracks. Since the first track is numbered zero, the highest one is number 34. So you want to start the extension process to start with track 35. After adding the tracks to the disk, Super Utility will ask you if it should rewrite the directory and boot. It is very important that you answer "no," since an affirmative answer would result in the current directory's being wiped out.

After the format is completed, finish the process by going to the Disk Repair module and running Repair GAT. Use a track-count override equal to the new number of tracks on the disk. In our current example, when Repair GAT asks 'Drive # ?', you'd respond '1A=40' (This procedure will give you a 40-track TRSDOS disk, but does not mean that TRSDOS will automatically recognize the extra tracks. TRSDOS for the Model I can be extremely stubborn about refusing to recognize that it has been handed a free gift --Ed.)

Here's something to beware of if you've hard configured your copy of Super Utility for a hex input default. If you try to enter a track number of D hex, Super Utility will interpret the "D" as indicating that you want the directory track. The way around this is to enter "0DH" instead of "D."

Earlier, I referred to the danger of using the double-step configuration for writing. To review, the double-step mode lets you use an 80 track drive (96 Tracks Per Inch) with diskettes formatted in 35 or 40 track drives (48 TPI). In such circumstances, it is safe to read. But if you try to write in

the double-step mode, the written track will not be as wide as a true 48 TPI track.

The write operation may seem to be successful at first. The trouble usually comes when and if the target diskette is placed back in a 48 TPI drive. The data which was written in the 96 TPI drive may no longer be legible. Because of this, it is recommended that you software write protect any drives that are configured to double-step.

In my own experience, I have gotten away with writing to disks while in the double-step mode. I have even been able to read those disks in my 35 track drive. However, there was one precaution I observed which probably contributed to my success.

The only disks I wrote to in the double-step mode were disks which were freshly bulk erased. Here's a recap of my formula. To begin with, I have one 35 track 48 TPI drive and two 80 track 96 TPI drives. Occasionally, I've needed to back up a 40 track 48 TPI disk. In such a backup operation, my 35 track drive can't handle the innermost five tracks of either the source or destination disk. So both disks had to be placed in my 80 trackers, both configured to double-step.

Before starting the backup, I bulk-erased the destination diskette. Then I performed the backup. The copy I ended up with seemed to work with 48 TPI drives. The bulk erase evidently ensured that there wouldn't be conflicting data streams under the wider 48 TPI read/write head.

The bulk eraser I used was Super Utility's Software Bulk Erase routine (in the Format menu), configured the drive for 80 tracks without double-step. Since this procedure worked for me, it may work for you. But don't count on it! Test the process thoroughly before entrusting valuable data to it.

If you use a product called the Patch, you may have some difficulty using Super Utility. the Patch is generally incompatible with any program which uses a debounce delay in its keyboard scan. If you have installed the Patch, you may have to install a patch that disables Super Utility's debounce-delay.

I once read something in Infoworld that stated that no disk-software protection scheme could work because whatever is read under head A can be simultaneously written under head B. By now you probably realize that with the TRS-80's FDC, this just isn't so. Thus, no program can automatically copy every protected disk. Human intervention may be needed to decipher the formatting tricks used and to set up a similar format on the destination disk. Super utility does an excellent job of attending to this chore automatically.

When it first came out, Super Utility's Special Backup could copy almost any software on the market. More convoluted protection schemes have evolved. Owning Super Utility no longer guarantees that you'll be able to backup any disk you buy.

Kim does not intend to improve Special Backup. Nowadays, most software venders have rational backup and replacement policies, similar to Kim's own. Such policies allow the user reasonable safety and convenience with regard to backups, and spare the programmers and vendors the gargantuan expense incurred by leaving themselves vulnerable to software pirates.

Chapter VII
ZAP, You're Dead!
Or, How Not to Destroy a Disk and Not Know It

Zap is probably Super Utility's most overwhelming module. It has the largest submenu and the greatest number of non-menu commands. It's also one of the easiest modules with which to cause irreparable damage, if you don't understand what you're doing. This chapter will put you and Zap through some paces together. If you follow the examples presented here, you'll explore all of Zap's features in a safe, step-by-step, fashion.

Let's start by creating a target disk (or maybe I should say a victim disk). Boot up Super Utility. Then remove it from drive zero and replace it with an unformatted disk, or a scrap disk. Configure the drive for Model I TRSDOS. In case you were absent the day I covered configuration, that means you have to go to Super Utility's configuration module, and enter the "A" DOS specifier for drive zero. Then go to Super Utility's Format program and format the disk in drive zero. Answer the 'Use configuration?' prompt with <Y><ENTER>. This will put a standard Model I TRSDOS format on the disk.

After you've formatted the disk, press <SHIFT><BREAK> to get the master menu. Then press <ENTER> to get the Zap menu. Look at the boot sector of the disk you've just formatted: press <ENTER> to select the Display Sectors option. Super Utility will prompt you with 'Drive,Track,Sector ?" Press <ENTER> again to default to 0,0,0. Super Utility should now read the disk's boot sector.

On the ASCII side of the display, you'll see the message, 'NOT A SYSTEM DISK' towards the bottom of the screen. This message would be displayed if you tried to boot the target disk. If you want to verify this, put the disk in drive zero and press reset. After you've seen the message, reboot Super Utility and get it to display the target disk's boot sector again. If you don't have ready access to a Model I, repeat the formatting process, but with a "D" DOS specifier, instead of "A." Use Zap to observe the 'NO SYSTEM' message and then try to boot the disk on a Model III. When you've satisfied your curiosity, Please reformat the disk using the "A" DOS specifier, so that you'll be able to follow the rest of this session.

Suppose we want to change the "NOT A SYSTEM DISK" message to something a little more dignified, like 'BOOT YOURSELF MAC'. The first thing to do is prepare for ASCII input. If you haven't changed your Super Utility defaults, you should currently be in

the hex input-mode. You can confirm this by noting the word "HEX" at the top of the display's lefthand column. You should also be in the Paging mode. This is indicated by the single blinking cursor above the word "hex." If you were in the modify mode, you'd see two cursors in different parts of the screen.

Change from the hex to the ASCII mode simply by pressing <A>. The word "HEX" should be replaced by "ASC" to indicate the new status. You could just as easily have entered the decimal, binary, or octal mode, by keying <D>, <B>, or <O>. Verify this by keying <D> and watching the input-mode designator change to 'DEC'.

Leave decimal as the input mode, for the moment, and go from the paging to the modify mode by typing <M>. The double cursors should appear. For the time being, we will be concerned mainly with the cursor on the ASCII side of the display. Use the arrow keys to place it over the "N" at the start of the message.

You'll notice a new field, called the cursor address field, has appeared above the input-mode designator. It should now contain the number '17.' Like many of Super Utility's displays, it is in hexadecimal. 23 would be the equivalent decimal value. The field contains the relative position of the cursors within the sector display. The first byte in the upper lefthand corner of the sector display is byte zero. If you count bytes from that corner, you'll find that the "N" (with the cursor over it) is indeed sector relative byte 23 decimal (17 hex). If you're count was off by one, you probably started counting with one instead of zero.

There are other ways you could have moved the cursor to the start of the message. Press <CLEAR>. This will "home" the cursor--move it to relative byte zero back at the upper lefthand corner of the data area. Now press <G>, for "GOTO." The cursor will change to arrows asking to be pointed in the proper direction. Send it to its destination by typing the digits 023. Presto! The cursor is back over byte 23 at the start of the target message.

An interesting point has come up here. Why was it necessary to type '023' instead of just '23'? It's because the decimal input mode requires three characters to specify a byte. Remember, a byte is an eight bit value ranging from 0 to 255 decimal. When you input a number in the modify mode, you always have to enter as many digits as it would take to express 255 in the current input mode. In other words, you need to input two hex digits (since FF hex=255 decimal), or three decimal digits (255 dec=255 dec) or three octal digits (FF hex=377 octal) or eight binary digits (FF hex=11111111 binary). You could circumvent this requirement, in a sense, by keying in fewer digits and then pressing <ENTER>. But since <ENTER> itself is a keystroke, there's not usually much to be gained by that approach.

Here's yet another way to move the cursor to the desired position. Use <CLEAR> to home it again. Now press <L> (for locate). Type 078. Again, the cursor goes where we want it. The trick is based on the fact that 78 decimal, or 4E hex, is the value of an ASCII "N." Verify this by looking under the hex cursor while the ASCII cursor is over the "N." So in asking Super Utility to locate 078, decimal, we sent it off after the "N."

One more time, please. Home the cursor. Now type <L><L>. Again, the cursor's over the same old place. Super Utility took the first "L" to stand for Locate. When you typed the second "L," Super Utility assumed it stood for Last. So it searched for the value used in the last search, 78 decimal or 4E hex. Naturally it ended up in the same place.

Now let's change the input-mode back to ASCII, without leaving the modify mode. Press <ENTER> while holding down <SHIFT>. You should see a cursor on the input-mode designator at the top of the display's lefthand column. Key <H> and you'll be back in the hex input-mode. Press <<SHIFT> <ENTER>><A> and you'll be back in the ASCII input-mode.

Now that you're using ASCII input, any alphanumeric key you press (including all punctuation and special characters) will be interpreted as ASCII input. Therefore you no longer have the <G> for Goto or <L> for Locate or Last features available. The <G> and <L> keys will now be interpreted as ASCII input.

Time to change that message. It would be a good idea to enter the new message in capital letters. To avoid the need to hold down the shift-key, engage Super Utility's CAPs lock by pressing <<SHIFT><ZERO>>. Now, type the new characters right over the old message. Both the old and the new message are 17 characters long. The old one has a period at the end; the new one doesn't. So don't type a period at the end of 'BOOT YOURSELF, MAC'.

While you were typing the new message, you may have noticed that the disk didn't spin. In case you're wondering how you can be modifying a disk without the drive turning, here's the answer. For the moment, you're only updating a buffer. The disk remains as is until you're positive that the information is correct as entered. If you make a mistake, use the arrows to put the ASCII cursor back over the error and retype the character. When you've finished entering the new message, press <ENTER>. You'll get the following modification menu:

U>pdate, R>eturn to modify, C>ancel ?

If you enter <U>, the changes you made on the video will be saved to the disk sector and become permanent (the sector will be Updated). Incidentally, hitting <ENTER> without any other input defaults to <U>. So don't press <ENTER> unless you're sure you want the sector modified.

Entering <C> will Cancel the changes you've indicated. The unmodified sector will be reread into the sector buffer, and you'll be returned back to Zap's paging mode.

Entering <R> will Return you to the modify mode. Your changes will not be saved to the disk, but they will remain in the buffer so you can continue modifying where you left off. <R> is useful for recovering when you accidentally hit <ENTER> in the modify mode.

To save the modified message we've created, type <U><ENTER> from the modification menu. That's all there is to it. If it seemed like a complicated process, it's only because we took many detours along the way, to play with special Zap features. If you want, test the modification now, by trying to boot the target disk.

There are still many Zap features left unexplored. Reboot Super Utility, go into Zap, and display the target disk's boot sector again. Go into the modify mode and put the cursors anywhere near the middle of the screen. Hold down the less-than key (<) for a moment, and watch what happens. This is like "delete" in Scripsit or the Electric Pencil. The character at the cursor is deleted and the characters to its right are moved over to fill the vacancy. The process ends at the end of the current sector. Zeros are "pulled in" to replace the deleted characters. There is no "wrap through" to the next sector. Only what you see on the video changes.

The greater-than key (>) is the inverse of the less-than key, corresponding to a word processor's insert character function. Each time you press it, all the characters to the right of the cursor are "moved over." The entire sector following the cursor is pushed to the right. A zero appears in the vacated slot under the cursor. The last character in the sector goes over the edge and is lost. Again, there is no wrap through to the next sector. What you see is what you get.

Unlike most other printable characters, "<" and ">" work even in the ASCII modify mode. So if you want to type in a greater or less-than sign as a disk modification, you'll have to change the input mode. If you go into the hex input mode (by pressing <<SHIFT><ENTER>> <H>), you can enter a less-than sign by typing 3C. A greater-than sign would be 3E.

In the next chapter, which deals with patching Super Utility itself, you'll get some practice modifying in hex, instead of ASCII. But first, let's look at some of the fancy options available in Zap's paging mode.

If you're still in the modify mode (two cursors), go to the paging mode by pressing <ENTER> to get to the modification menu, and then <C><ENTER> to complete the journey back to paging. If you've been staying with me, you're now in the paging mode, still looking at track zero, sector zero of the victim disk. Tap the right arrow key. The display should advance to the next highest numbered sector: sector one. The left-arrow key works

the same as the right-arrow key, except that it moves you to the next lowest sector, instead of the next highest. Use it to step back to sector zero.

Now press <<SHIFT><right-arrow>>. As you see, the right arrow seems to work the same way, shifted or unshifted. The difference becomes apparent at a track's highest or lowest numbered sector. These keys auto repeat, so hold down <<SHIFT><right-arrow>> and watch the sectors skip through the display. The last sector you'll be able to read that way will be the highest numbered sector on the track, sector nine. When you attempt to pass it, Super Utility will give you a 'Disk Read Error, Sector not Found' type message. This is a handy warning that you've come to the end of the track.

Answer the mini-menu by entering <S> for S>kip. When you find yourself back in the paging mode, use the left arrow to back up a sector or three. Now advance again, this time using the right-arrow without the shift-key. After the track's last sector has been loaded, the display will advance to the lowest numbered sector of the next highest track.

The difference is the same for the left-arrow versus shifted left-arrow keys. The unshifted arrow is usually more convenient to use. But if you want to restrict yourself to a certain track, use the shifted arrows.

If you want to jump several sectors (e.g. from sector zero to five), you don't have to step sector by sector. Just press the number key which matches the sector you want to go to. For instance, to go directly to sector five of the current track, just press <5>. If you have double density, you may be wondering what to do if you want to jump to any of the sectors from 10 through 17. To do so, press <S> for Sector. Super Utility will then prompt you to enter the number of the sector you want displayed.

The functions of the up- and down-arrows are similar to those of the left and right arrow, except that the track changes, instead of the sector. The shifted up- and down-arrows take you to the disk's highest or lowest configured track, respectively. To jump to a track without using an arrow key to single step, press <T> for Track. Super Utility will prompt you to enter the track and sector number you wish to examine. If you want to look at a disk in another drive, press <CLEAR> and you will be prompted to enter the drive, track, and sector number.

There is yet another way to step from sector to sector. The greater-than (>)and less-than (<) key work similarly to the right and left arrow keys. There is a difference, but to demonstrate it, we'll have to create a special track on the victim disk.

Go to Super Utility's format module again. Select the Build Format Track option. Answer the DOS specifier prompt with A. Answer the Track prompt with five. When you're invited to press <ENTER> to see buffer, do so. You will then be in the memory zap

mode. Make sure the input mode is hex. Then Press enter to go into the memory modify mode.

Type <L> (for "Locate,"). Then type FE. The cursor will position itself over the first FE in the sector. As described elsewhere in this book, FE is the ID address mark which declares the start of a sector header subfield. The first byte after the FE should be an 05, indicating the target sector is on track five. The next byte, the header byte, should be 00. The following byte is the sector number. Super Utility will not always start the track with the same sector. But if this is the first format track you've created since booting up Super Utility, the sector byte should be 08, indicating sector number eight.

Place the cursor over the sector byte. Now change it to 40 by simply by typing 40 over the 08. What we've done is to create a memory image of a nearly normally formatted track. The only reason for the "nearly" is that it has no sector eight. Instead, it has a sector 64 (40 hex=64 decimal). The next step is to put that memory image onto an actual track.

Press <BREAK> to get back to the Format menu. Select option 5, Write Format Track. Answer the drive and track prompt with the drive number or the target disk and track five. The result should be a sector eight-less track five on the target disk.

Press <<SHIFT><BREAK>> and go back to the Zap menu. Display track five, sector zero. Now hold down either the right-arrow key or the shifted right-arrow key and let Super Utility zip through the sectors. After sector seven, Super Utility will stick for a moment, and display a sector not found error. This is natural enough, since it's looking for sector eight and there's no such sector.

Answer the mini-menu prompt with <S> for S>kip. Then use the left arrow to back up a few sectors. Now start your approach again, this time using the greater-than key (>) to step through the sectors. As you see, Super Utility now has no trouble at all finding sector nine, which is the next configured sector after seven. After nine, the greater-than key will step Super Utility to sector 64, which is the track's next highest configured sector. To sum up, the greater-than key finds the next highest configured sector on a track, regardless of the numbering scheme. The less-than key (<) works similarly, but finds the next lowest sector.

****

PATCHING Super Utility

You should now be thoroughly familiar with Super Utility's Zap mode. You've used it to customize a boot. You've even created a nonstandard-format track with it. By now you should be

ready for what may be the ultimate zaplication: Zapping the Super Utility disk itself!

Recently a set of patches to Super Utility was issued by Breeze/QSD Inc. to correct some of the bugs in the 2.2z version of Super Utility. The rest of this section deals with installing such patches. Even if you were lucky enough to have received a copy of Super Utility that was already patched, this section may still be worth going through.

If you haven't already, please read the section of the Super Utility manual labeled patch sectors. You'll find it starts with a list of TRSDOS standard Load File Format Codes (LFFC's). The treatment of the subject is a little sketchy, so here's a fuller explanation:

There are two kinds of files. The first must load to a specific area of memory. The most important example of this type of file would be a location-dependent machine language program, which must load to the address at which it's meant to execute. Any file with a /CMD extension is probably one of these location-dependent files.

The second type of file has no special memory location associated with it. Data files are usually of this ilk. For instance, a word processing file may be loaded into whatever text buffer is used by the wordprocessor. BASIC programs are also non-memory specific. In a sense, a BASIC program is data used by the BASIC interpreter. Different versions of BASIC may have differently located text buffers into which a program is loaded. The program will be just as happy in any location.

For the purposes of patching Super Utility, we are primarily interested in the first type of file--the one which must reside in a specific memory location. Such a file is said to be a load file. A given load file may contain several separate load modules, each of which loads to a different area of memory. Load modules usually contain information which tells the system loader where in memory the module belongs. There is a standardized way in which such load-address information is included in the file. Load files conforming to that standard are said to be in load file format.

Load file format uses certain codes to separate data which is part of a load module from information which tells the system loader how to handle the load module. Those codes are called load file format codes.

The LDOS Quarterly, April 1, 1982, contains a detailed essay on load file formats and the load file format codes (LFFC). The Super Utility manual refers to four kinds of standard TRSDOS LFFC. "00" is a termination code. It signifies the end of a load file. "01" is a load marker. It signifies the beginning of a load module. "02" is an entry point marker. It is used by DOS, but has no application to the current generation of Super Utility patches. The 29 integers from 03 hex to 1F hex are

identical LFFC's, as far as Super Utility is concerned. They indicate a remark--an part of a load file which doesn't actually load into memory at all. Such remarks are most commonly used for copyright messages, dates, etc.

Let's look at a typical load file:

0108003C4C494E452031010B403C4E455854204C494E4502022D4000

For the sake of readability, I'll repeat the above with spaces between the bytes:

01 08 00 3C 4C 49 4E 45 20 31 01 0B 40 3C 4E 45 58 54 20 4C
49 4E 45 02 02 2D 40 00

The 01 at the start of the sequence is a LFFC marking the beginning of a load module. The first byte after an 01 load marker is always a length byte containing the number of bytes in the module. In this case, the byte is an 08, indicating an eight byte long load module.

The pair of bytes following the length byte always contains the address to which the first byte in the module loads. As is normal with Z-80 code, the address is in Least Significant Byte, Most Significant Byte format (LSB,MSB). In our example, the two bytes after the length byte are 00 3C indicating a load address of 3C00 hex. Hmm, that's the start of video memory, isn't it? I bet this module is some sort of message!

We said that the length of the module, as indicated by the length byte, was eight. The address field always uses up two bytes of the specified length. So there are six bytes left for our actual load module data. In general, a load module's data segment will be two bytes shorter than the length indicated by the length byte, because of the two byte address.

The system loader has enough sense to subtract two from the length byte to derive the correct load module length. This subtraction is an eight bit operation, without carry. This means that if the specified length is two or less, the result "wraps around." In other words, you may specify a true length of 256 by using 02 for the length byte. Using 01 will result in an actual length of 255, and 00 will result in a true length of 254.

In our current example, the six bytes of load module data are 4C 49 4E 45 20 31. These just happen to be the ASCII equivalents of 'LINE 1'. So what we have so far is a message which loads to the first line of the video which says 'LINE 1'.

After the sequence 0108, the system loader knows that there won't be another LFFC for eight bytes. Therefore, the zero in the address field didn't confuse it. The system loader interpreted the zero as part of an address, not as another LFFC. There could have been 00's, 01's, and 02's anywhere within the specified eight bytes, without causing any confusion. But the next byte, after the eight had better be another LFFC. In our

example, it's an 01, indicating the start of another load module.

The length byte for the new module is 0B hex, indicating a length of 11 (decimal) bytes. The address field contains 403C. This points to memory location 3C40 which is the start of the second video line. After the two address bytes are subtracted from the length of 11, there should be nine additional bytes of data. Nine there are: 4E 45 58 54 20 4C 49 4E 45. In ASCII, this is 'NEXT LINE'. We have used up the 11 byte specified by this module length byte. So the next byte, once again, should be a LFFC. Sure enough, it's an 02. This indicates a load file entry point follows. The next byte is a length byte indicating that the address field will be two bytes long. This comes as no great shock to the system since Z-80 absolute addresses are always two bytes long. The address bytes themselves are 2D and 40, indicating the address 402D which is the normal DOS READY entry point.

Next is the termination LFFC, 00, which indicates the end of this load file. If an operating system such as TRSDOS loaded the file we have been describing, the net result would be to print the message 'LINE 1' on the first video line, and 'NEXT LINE' at the second video line and jump to the DOS READY state.

You now have most of the background needed to understand the system of Super Utility patches. Another important thing to realize is that the entire group of Super Utility patches constitute a single load file. Each individual patch is a load module (or group of load modules) within that file.

If you read the information on the Super Utility patch sheet, you'll come across Kim's warning to those of us who don't apply all the patches in one sitting. It specifies that we mustn't leave any zero bytes between patches. Now that you know that all the patches are treated as one load file, you should see why the presence of an extra 00 where the system is anticipating a LFFC would terminate the loading process.

I also urge you to heed Kim's other suggestions, especially the ones about checking your work very carefully, saving your progress to disk frequently (when I installed the patches on my copy of Super Utility, I saved my work after every patch), and using a pencil to cross out (or underline) each patch as it's applied.

Let me go through the patching process with you:

Remove the write protect tab from your Super Utility disk. Boot up Super Utility (Hold down the CLEAR key to establish the version number). Make sure the version number displayed is 2.2z. Press <ENTER> to go into Zap. Press <ENTER> again to select the Display Sectors mode.

MODEL I
Answer the Drive,Track,Sector prompt with 0A,1,6<ENTER>.

MODEL III
Answer the Drive,Track,Sector prompt with 0b,2,6<ENTER>.

ALL USERS
Please note: the only difference between the Model I and III procedure so far has been the DOS specifier ("A" versus "B,") and the track number of the patch sector (1 versus 2).

If Super Utility isn't already in the hex input mode. Press <H>.

Press <M> to get into the modify mode. The cursor should be over relative byte zero. If you haven't already applied any patches, that byte should be a 00. There should be a message in the middle of the sector stating that the sector is indeed a patch sector. The message may be an annoyance when you're patching over those bytes, so clear the sector by pressing <P> (for propagate). The cursor should enlarge. Type FF. The byte under the cursor (00) should be copied (propagated) to all 256 (FF hex=256 decimal) of the sector. This will erase the message.

We might as well save what we've accomplished, so press <ENTER>. You may answer the next prompt with <<U><ENTER>>, or simply <ENTER>. When the sector has been saved, Zap will be back in the paging mode, so press <M> to return to the modify mode.

Again, the cursor should be over relative byte zero. Remember, we're primarily concerned with the hex side of the display, not the ASCII side. Time to enter the first patch.

MODEL I
Type 01 05 AC 7C 00 00 00

MODEL III
Type 01 05 8C 7C 00 00 00.

ALL USERS
Save this patch by pressing <ENTER> twice. Then use a pencil to put a line through the patch on the patch sheet.

Let's look at what we've entered. The first byte was an 01. This is marks the beginning of a load module. The following byte indicates a length of five bytes. The next two bytes indicates the starting address of the memory block to which the load module is to be loaded. The address is in MSB,LSB format. The Model I address is 7CAC hex. The Model III address is 7C8C. After the address, there are three bytes left out of our five byte length. These three bytes constitute the actual body of the patch. They are identical for both models: 00 00 00. It looks like Kim is just replacing a few machine language instructions with NOP's.

Time for the second patch, which, you'll notice, consists of two "patchlets." Re-enter the modify mode. Now position the cursor after the last byte of patch one. That should put the cursor at relative byte seven.

MODEL I
Type 01 05 40 C1 CD 77 9E

MODEL III
Type 01 05 0C C1 CD 49 9E

ALL USERS
Save the patchlet by pressing <ENTER> twice and cross it out on the patch sheet. Press <M> to re-enter the modification mode, and place the cursor on the next byte after the 9E.

MODEL I
Type 01 05 47 C1 00 00 00

MODEL III
Type 01 05 13 C1 00 00 00

ALL USERS
Save the patchlet by pressing <ENTER> twice and cross it out on the patch sheet.

Suppose you decided you'd had enough patching for one session. Re-enter the modify mode. Find the last byte of the last patch installed. That should be the third zero after the C1, or sector relative byte 14. You have to leave an additional zero byte after the last patch so that the system loader (in this case Super Utility itself) will know that the end of the load file has been reached. So place the cursor over relative byte 15 and make sure it's zero.

Next, place the cursor over relative byte 16. Switch to ASCII input by typing <<SHIFT><ENTER>><A>. Leave yourself a message by typing "Start next patch at relative byte 15."

Note that you start at 15 because when subsequent patches are added, the termination mark (00) currently at byte 15 will no longer be appropriate. Use the old press <ENTER> twice trick to save the message, and you're all set to remove your Super Utility disk and relax. By the way, I recommend your keeping a write protect tab on you Super Utility disk when you're not actually patching.

When you're ready to patch some more, remove the write protect tab, boot up Super Utility, go into Zap and redisplay the patch sector. Enter the modify mode and place the cursor over the byte which your message says is the place to resume (byte 15).

Type in the next patchlet: 01 05 6A 41 .... Continue following the instructions on the patch sheet. When you get up to patch eight, you'll be directed to switch to a new track. Before doing so, be sure to save the work you've entered on the current sector.

You'll be given a procedure which involves typing in two entire boot sectors, and then loading the sectors to memory and dumping them to other sectors. But it may not be necessary for you to do all that. The boot sectors provided by Super Utility are not compatible with every DOS, only with TRSDOS and similar operating systems. The hard configure section of your Super Utility manual tells you how to copy the boot sector of your favorite DOS onto Super Utility's repair Boot boot. If you plan to do so (or have done so), you needn't bother typing in the two boot sectors in the patch sheet. Of course, you can also skip the part involving the memory utility.

If you do type in the new boots, save your work frequently. Whether you type them in or not, you still must apply the part of patch eight which goes into the patch sector. It consists of five patchlets and may be found just before patch nine.

After the twelfth (and last) patch has been entered and saved, don't forget to put in the terminating 00 byte, and a message such as the one in Kim's example. Remember to replace the write protect tab.

Check all your work very carefully. If you've made an error and you don't catch it, the next time you boot up, Super Utility may go out to lunch and stay there. If that should happen, don't get upset. Just reboot Super Utility and hold down the <CLEAR> key until the program is completely loaded. That will prevent the patch sectors from loading, and you'll be able to use Zap to correct the error(s).

N O T E S

Note 1

Zap's "Read ID Address Marks" function has two modes. In mode one, the last three columns of data (headed "CKCRC," "IBM," and "DATA) are not displayed, and in mode two, they are. You may toggle between modes by holding down the <X> key until the mode changes.

Though mode two has more information than mode one, for most purposes, mode one is preferable. The reason is that it works more quickly, and a certain disk r.p.m/interrupt phase lock problem is less likely to develop.

This phase lock problem has to do with the fact that an accurately timed disk drive spins at the rate of 300 revolutions per minute, or five times per second. TRS-80 interrupts occur 40 times per second. Note that 5, the number of disk revolutions per second, is an exact devisor of 40, the number of interrupts per second.

This relationship can cause problems which have plagued many TRS-80 owners. It is responsible for some of the "silent deaths" or "time-outs" which darken the lives of TRS-80 users. Super Utility is fairly immune to this difficulty, but it can slow things down when using "Read ID Address Marks" in mode two. It can also make it difficult for mode two to find all the sectors on a track.

One way to help over come this problem is to break the phase lock by pressing <SPACE> to freeze the action and letting the disk stop turning. Then restart it by pressing <ENTER>.

Read ID Address Marks's default mode is mode one. Mode one is not prone to phase lock problems.

Note 2

The up and down arrow keys have auto repeat in Zap's "Read ID Address Mark" function. However, auto repeat is very slow in mode 2 (the mode in which the three right-most columns of data are displayed (see note one). Auto repeat is also slow when Super Utility has trouble reading the ID marks. In either of the above cases, you will do better to tap the arrow key repeatedly, instead of holding it down.

Note 3

Super Utility has a "mini-menu" which is invoked whenever a disk I/O error is encountered. At such times, you will usually see a description of the error condition followed by the mini-menu. It should look something like this:

```
Disk READ ERROR !
Drive 0, Track 01, Sector 01.
Sector NOT FOUND !
R>etry, S>kip, C>ontinuous, N>onstop, Q>uit ?
```

If you answer the prompt with <R>, Super Utility will try the disk I/O operation once more. If it succeeds, it will continue with what ever function was under way when the error occured. If it fails, you will be returned to the mini-menu.

Selecting <S> from the mini-menu will skip the problem sector, causing Super Utility to resume the interrupted function at the next sector.

Selecting <C> is like selecting <R> repeatedly. After <C> has been entered, Super Utility will go into an indefinite retry loop which will continue until the disk I/O succeeds, or you interrupt it with the <CLEAR> key. If you press <CLEAR>, you will be returned to the mini-menu. If, on the other hand, Super Utility eventually succeeds with the difficult I/O, it will continue with the interrupted function. If another problem sector is encountered, Super Utility will return you to the mini-menu.

Selecting <N> from the mini-menu is like selecting <C>. Super Utility will automatically retry until it succeeds or is interrupted. However, selecting <N> also causes the mini-menu to be bypassed on future I/O errors. When such errors occur, Super Utility will automatically go into the continuous retry mode. If Super Utility gets stuck on an unreadable sector, interrupt it with the <CLEAR> key. You may then use the <S> option, which will cause Super Utility to skip the impossible sector and resume its task. Choosing the <S> option at this time will also cause Super Utility to revert to prompting you with the mini-menu whenever an error occurs.

All this has been covered in your Super Utility manual. The only reason I'm rehashing it here is to bring up the following point:

BEGINNING OF THE MAIN POINT OF THIS NOTE:

If you choose <N> from the mini-menu prompt, it will take you out of the mini-menu-prompt-mode. Super Utility will remember this mode change until you reverse it by pressing clear and entering <S> to skip a sector. If you don't do this, you may

later use another Super Utility function and, upon encountering an error, be thrown into a retry loop. For instance, you might use the <N> option during a Sector Verify. Then you might perform a number of different Super Utility operations. If you don't encounter any I/O errors, you may forget you're in the no-prompt mode. 20 minutes after the Sector Verify, you might try to read a sector with Zap. Suddenly, the disk starts repeating the same operation over and over again, and an error message flashes on the screen and disappears.

THE MAIN PART OF THE MAIN POINT OF THIS NOTE:

For this reason, sometimes when this manual says you will get the mini-menu, you won't. Instead, you'll go into the automatic retry loop. When this happens, simply wait a few seconds to give Super Utility a chance to successfully complete the operation. If it continues to fail, just press <CLEAR> to get the mini-menu prompt, and then enter <S> to skip the problem sector and revert to the standard mode.

Afterword:
On Becoming an Expert

by Kim Watt

First off, I would like to express my appreciation to Paul Wiener for his professional work. He has provided SU+ users with a great piece of art which is technical and novice both in the same breath. During some of those mind stretching moments that we all had in reading this manual, Paul has lightened things up a bit allowing us to take a deep breath and absorb it all in. Also, I would like to thank Dr. Renato Reyes for his technical coordination and editing, and Dennis Brent for his expertise in getting the project completed and out to the users. All in all, this manual is very complete in every aspect, and SU+ users who started out as novices before reading this manual, should be 'intermediate' users by this time.

As an SU+ user myself, I would like in this section to explain to you some of the shorthand that I use myself.

The SU+ program, as we all know, is very huge and very complicated. But by using the menu structures, complicated commands may be passed to the program easily without having to memorize lengthy rules of syntax. This is what allows even the beginning SU+ user to manipulate data in a sophisticated manner. A new user who boots up SU+ for the very first time really doesn't realize just how much power has been made available to him. After reading the manual, and playing with the program for awhile, the user moves his skill level up to the novice category. If the user is very familiar with the TRS80 and disk systems in general, he can easily move himself up from there to the intermediate level. Some of us, however, need more time to reach this level. We thirst for knowledge, not being satisfied with knowing that pressing this button will repair a GAT table, we want to know WHY, and by the way, just what the hell is a GAT table anyway?!

In reading this manual, those of us who are not real familiar with the machine we are using will have learned a lot of material. So at this time I would like to take the next step forward and help everyone to be an expert SU+ operator.

The first thing to do in order to become an 'expert' SU+ user is to first become a 'novice', and then an 'intermediate' user. During these first two phases you will master most aspects of the program, and especially, memorize the DOS symbol list of a-j. You will also know just what disk type A is, what type B is etc. Some of the symbols will define a similar diskette structure. The A, C and E are identical (single density), the B is unique (double density sectors 1-18), the D,

G and J are identical (double density sectors 0-17), and the F, H, and I are identical (double density, single density track 0). The only differences that arise within these groups are intimacies within the file structure itself, and not within the diskette structure. This means that by using the specifiers A, B, D and F only, we can define the entire range of standard diskette structures. We can use these symbols on any disk as it applies, as long as we don't need any file handling support. This would mostly be applicable for 'zapping' only, but if you are like me, this is the main feature of SU+ that you use the most. In summary, if you know you are going to look at a single density disk use A, if double density, use D, etc.

Another point to make about the use of these symbols is that none of them will invoke SU+ to use 'disk relative sectors (DRS)' as its addressing scheme. By using the above suggested symbols, you will know that the sector you asked to read is the one that you see, and not a 'computed (relative)' sector. This will always allow you to work in a physical sector environment (versus logical sector).

A lot was mentioned in this manual on how to configure the SU+ system, but let me tell you how to do it in 'expert' mode. To be an expert, you must go ahead and trust yourself and SU+ enough to HARD CONFIGURE (gasp!) your SU+ diskette. I know that many of you have refused to take off the write protect tab on your master disk, so I am giving y'all my permission to go ahead and do it right now. Go right ahead and hard configure like a madman. Make everything just right for the way that you normally use SU+ using instructions wherever you may find them. Don't worry about things like where the directory is (unless ALL your disks are the same), and what type of DOS you have set. What you want to do is tell SU+ how fast your drives can go, how long to wait on first disk access, and if the drive is actually in the system or not. If you don't have 4 drives, mark them with a minus symbol (-) as the first character. SU+ won't bother with them anymore. Then go ahead and write that data back on the disk.

Now, that wasn't so bad after all was it!?

Everytime you boot in your disk from now on things will be just the way you want. Go ahead and try right now.

One thing that I should mention at this time is that an expert always runs his SU+ with 'save config = N'. This is one key to quick data access. This feature was placed into the program to allow the first time users to set it and forget about it, but we're not novice anymore right?

At this point, every time you boot up SU+, it will know exactly where everything is, and how fast to make it go. Once you have done this, you will not ever again need to 'soft configure' the SU+ program. I can't think of a single thing in

that feature of the program that I would need to set that I haven't already, or couldn't do at any time. This will save a little time whenever you use the SU+ program. No more need to configure, just jump right over to the section that you need and go, go, go.

Since we are running with 'save config' off, each time we change disk types, we will want to reconfigure by using the DOS specifier after the drive number. The first time we look at a single density 40 track disk in drive 3 we would type "3a=40". The next time we would only use the 3 for the drive number because we have instructed SU+ to remember just what it is in that drive 3, and not to revert back to a previous configuration. If we now want to look at a mod III TRSDOS disk, we would just enter "3b", and only on the first time we access the drive.

SU+ has default conditions spread throughout the program's operation. Use these defaults to their advantage. We know that drive 0 is the default for every drive prompt, so why don't we just go ahead and use drive 0 as our standard. This will force us also to remove the SU+ diskette from that drive, and tuck it safely away for the next time it is needed. Using this technique, from the main menu, we can just press <ENTER> and hold it down, and the repeating keyboard will take us right into the zap menu, and answer prompts and display track 0 sector 0 of the disk. No need to select the menu options by number if it is the first one. Remember, SU+ always defaults to the first condition displayed on the video. <ENTER> will go to the zap menu just as fast as <1> <ENTER>, but with half of the keystrokes. If we had wanted to zap drive 3 instead, we could have hit the <ENTER> key twice from the main menu, and entered <3> <ENTER> at the resulting prompt. Likewise, if you are zapping, and are in the modify mode, and wish to write the sector back to the disk, just press the <ENTER> key twice.

Learning these things will help you speed up your throughput on the program's operations.

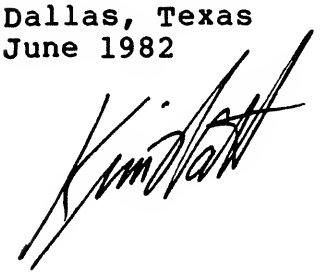
Many techniques and theories have been offered on how to recover data from a diskette that has gone sour on us. All of the discussions have been very thorough, and some very enlightening. I will now give away a big secret on diskette recovery. All you have to do is back the bad diskette up to a good diskette using the standard disk backup feature in SU+ along with the correct DOS specifier. Use retries heavily as disk errors come up. Believe it or not, this simple procedure will get everything off of the bad disk that is going to come off. All of the correctable CRC errors should be corrected during this process, and if that was the only type of error encountered, you should have a fairly accurate copy at the end of it. Now you can give SU+ a run for its money by read protecting the directory, and repairing the HIT and GAT sectors.

If all of this doesn't work, you can just about kiss those files goodbye, or else hire a professional to muck through your disk and re-enter the directory by hand. Enough just cannot be said about making backups of your valued disk files. SU+ will come in handy when your originals fail and you have no backups, but there is no reason for you not to have a backup. SU+ (and you) would much rather be copying one good disk to another than trying to make some sense out of a jumbled disk.

When you have recovered data from the bad disk, don't try to re-use the disk like it is. Just go ahead and reformat it after copying off all of the files. If it appears flaky during formatting, don't use that disk anymore. I don't feel at all bad about throwing a $5.00 diskette in the trash can when I think of myself punching holes in the wall later when an entire day's work suddenly disappears to NOT FOUND messages. Always use high quality diskettes from a reputable manufacturer. You spent all that money on the computer, don't go feeding it dog food now.

The final step in becoming the 'expert' user of SU+ is to use it all the time. Nothing is like hands on experience. We could publish three books a week on SU+ and never convey the same experience as being right there pushing the buttons. Go ahead and get a feel for how everything works. Back up a few diskettes and practice on them. If you make a mistake, you will not hurt anything, and you will get a good feel for what you and SU+ can do together. Hopefully you will make a good team!

Dallas, Texas
June 1982



Parting Words...

INSIDE SUPER UTILITY PLUS

or

"SU+... and NO MORE Mysteries!"

INSIDE SUPER UTILITY PLUS was a LONG time coming. The program, SU+, is SO comprehensive in its structure and design, that it was an ENORMOUS job to finish it! Not only that, but there are many UNDOCUMENTED features that ALSO get documented here. First, a little background....

Super Utility. That name brings up visions of magic. A product that takes a name like that BETTER be DAMN good! I first heard of Super Utility (the first version) in October, 1980. Victor Andrews, of Soft Sector Marketing, sold me my first personal copy (Serial #72) at the Boston Computer Show. I had never heard of Kim Watt, and I'm sure most of you hadn't either. FIFTY DOLLARS for a program! I thought it was a lot of money, considering that back then a DOS was $99 and programs had not reached that price level yet, as a rule. I traded Vic a copy of VTOS 4.0.2 for it, so I figured that wasn't too bad! (note: in the software biz, we trade software too... except they are always legal copies with a registration card! We love it, though!) I took SU back to Dallas for evaluation for inclusion in my Quality Software Distributors Catalogue. I booted it up, played with it, and thought, "My God! This is too much power for the average user!" That, of course was the feeling of many software houses at the time, as Super Utility had a reputation for copying ANY disk. I was really pleased when people started realizing that it had OTHER purposes! FORMAT WITHOUT ERASE was always a clincher when giving a demo. If you could find just a few features that were for YOU, then SU was worth buying, besides for a COPY utility. I loved getting calls saying how great the program was. Some guy's 12 year old kid's life was spared when a crashed disk was repaired. Poor kid turned off the computer with the drive door closed. Daddy's VISICALC disk was still in the drive from his all night session right before tax time. When he tried to BOOT the disk, strange enough, it wouldn't! When he put the disk in drive :1 to see if he could access it from there, he got a GAT error! WOW! No backup, of course. (It was 3:30 in the morning when he finished, damn it! Sound familiar?) Well, to make a long nightmare short... SU to the rescue!! YEA!!

Breeze Computing then came up with Super Utility Plus. In this one, Kim concentrated on getting not only single and double density working, but ALL the popular DOSes, and while he was at it, Mod I and III on the same disk! The copying of protected disks was NOT improved, therefore MANY of the game diskettes sold now are NOT copyable by SU or SU+. This is not an avenue that Kim pursued. It would be a pointless game of chase between Kim and the game whiz-kids. Instead, utilities were stressed and they were made workable with all systems.

Kim Watt. What a genius! This guy is SO good on these damn things! To watch him program is like watching a master at his craft. There are only a few as good as Kim, and not many better. I first met Kim in New York at the first (and I think LAST) 80 MicroComputing TRS-80 Show at the Statler. (The following year it was an Apple/80 show, and Wayne had no direct involvment). Anyway, here's this skinny blonde kid, around 6 feet tall, wearing silver Ray-Ban sunglasses, driving a royal blue Formula Firebird that says he's Kim Watt! Can you believe it? Oh well, he sounds like he does on the phone, so I guess it's him. We get along real well and have some fun in the Big Apple. Four months later Kim is moving from Detroit to Dallas for good and Breeze/QSD, Inc. is born. One month later, Kim has boots, belt, and cowboy hat. Detroit?? Where's that?

Super Uilility Plus is being written for the IBM PC, and will also be available on many Z-80 machines in the future. Model II/16 users, you won't be left out either! Kim is involved in many projects at the moment, and you will be seeing his name pop up in quite a few places.

I would like to thank Paul Weiner, first of all. Paul wrote a review of Super Utility and Super Utility Plus that was published in the January 1982 80-MICROCOMPUTING. It was so comprehensive and so well thought out that Paul was contacted for writing this book.

Dr. Renato Reyes, Head of our Technical Department, read Paul's chapters as they came in, worked with him, and did the final editing. Kim Watt read the final draft looking for errors. I read THAT final copy, looking for anything else that didn't look right to me. Hopefully, the result is a book that will be quite useful to you.

Lastly, I would like to thank Kim for being not only the best programmer I have ever seen, but a good friend as well.